Cisco Virtual Topology System (VTS) 2.3 Developer Guide

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

Abstract
The Cisco Virtual Topology System (VTS) 2.3 Developer Guide gives information on VTS APIs and development features.

The Virtual Topology System (VTS) provides L2 and L3 connectivity to tenant, router, and service VMs. The main components of the VTS are the Virtual Topology Controller (VTC) and the XRVR.

Cisco Virtual Topology System 2.3 Developer Guide

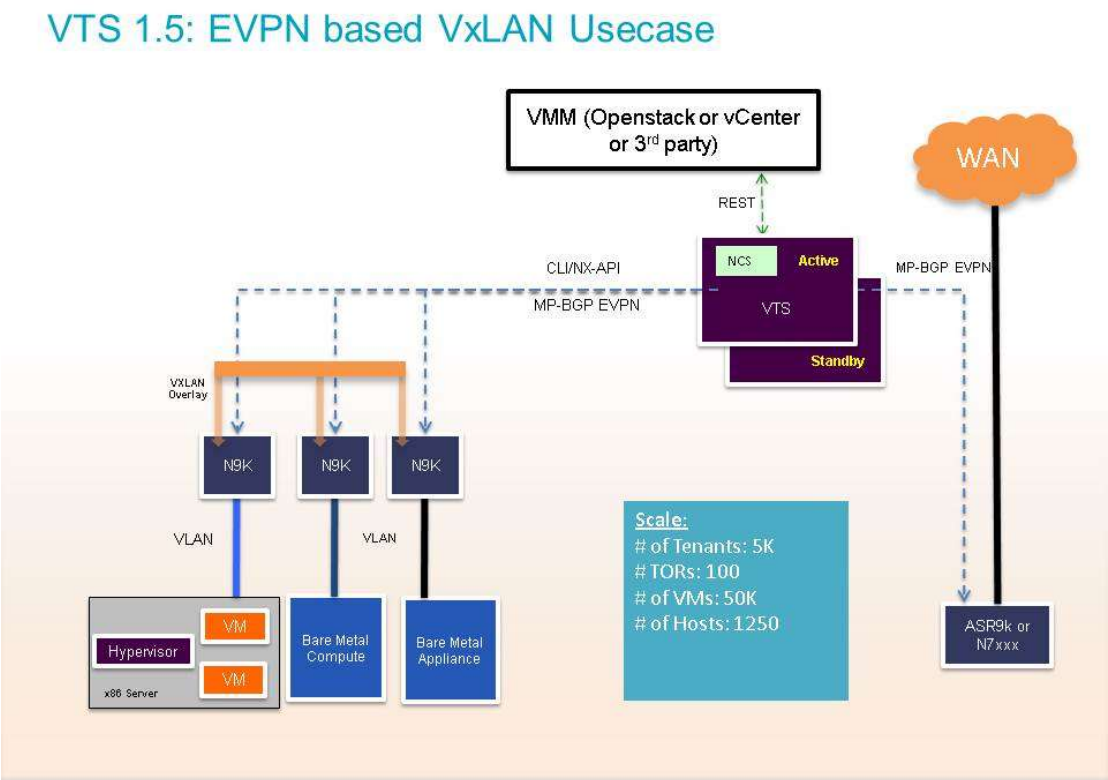
1 Contents

2 Overview of VTS 2.3 API

The VTS 2.1 API is exposed north-bound by the VTC component. The API is used to:

- Provision VTC with the Inventory of the DC topology
- Provision VTC with the Hardware VTEP Info
- Provision VTC with the information about the networks that will be serviced by the associated tenants
- Provision VTC with the information about sub-networks
- Provision VTC with the information about VMs attached to the networks
- Provision VTC with the information of L3 router and its interfaces that will be used for L3 connectivity between the tenant VMs

2.1 Architecture



2.1.1 Virtual Topology Controller (VTC)

VTC is the topology controller for the VTS; VTC controls all the hardware VTEPs to provide intra DC connectivity between VMs belonging to the same tenant zone in the DC and inter DC/WAN connectivity to remote networks belonging to the same tenant. VTC learns about remote networks by peering using BGP and other routing protocols to remote peers and route reflectors.

The VTC VM will have one VNIC interface which has VIP address:

- One for control, used in the NB API to program VTC.

2.2 Key Concepts

The VTC northbound API is a REST API over HTTPS. A VM managing entity is assumed to create VMs and call the VTC APIs to provide the required information for the creation of the virtual networks.
For security reasons the API implements certificate-based validation both on client side (the Managing Entity) and server side (VTC). The user is responsible for creating and uploading the server-side certificate in VTC, as well as creating and using the certificate on the client side. Instructions for the upload of certificate are specified in section 4.4.

3 Preparing to Use the VTS 2.3 API

The VTC NB API is used to provision the VTC with all hardware VTEPS that it controls, as well the networks, subnets and VMs that will be deployed on the compute servers.

3.1 Requirements

In order to exercise the API you should be able to issue https requests to VTC. This can be done by a binary or script. This document provides client-side examples using the curl command.

3.2 Installing the VTS 2.3 API

The VTS 2.3 API uses server-side certificate for security. The VTS 2.3 comes with default server-side certificate. The user is responsible for creating and replacing such certificates if needed.

3.2.1 Installing Certificates

- Key-file: Specifies which file that contains the private key for the certificate
  Copy the key file to /etc/ncs/ssl/cert/host.key

- Cert-file: Specifies which file that contains the server certificate.
  Copy the cert file to /etc/ncs/ssl/cert/host.cert

- restart the VTS server process:
  `sudo service ncs restart`

3.2.2 REST Semantics

- ***Retrieving a Resource***

GET is used to retrieve a representation of a known resource.
The XML representation has the list of container name as the XML document root.
Request query parameters
"deep" : retrieve a resource with all subresources incline
"shallow" : retrieve a resource with no subresources incline
"offset"/"limit" : used to specify a limited set of list entries to retrieve
"select" : used to select which nodes and subresources in a resource to retrieve

- ***Replace a resource***
  PUT is used to completely replace a known resource.
- ***Update some properties of a resource***
  PATCH RFC 5789 is used to edit a known resource. PATCH cannot be used to change keys of a resource.
- ***Create a Resource***
  POST to the parent resource to create a new resource.
- ***DELETE a Resource***
  DELETE is used to remove a known resource. DELETE can be used to remove the entire resource or part of it.

4 API Reference

Base URI

https://<vtc-ip>:8888/api/running/cisco-vts/

Headers:

Content-Type:

application/vnd.yang.data+json **OR** application/vnd.yang.data+xml

Accept:

application/vnd.yang.data+json **OR** application/vnd.yang.data+xml

Accept while retrieving a list of resources:

application/vnd.yang.collection+xml **OR** application/vnd.yang.collection+json

Authorization:

Basic: (username/password)

4.1 Understanding VTS Payloads:

GET a resource:

Response:

- The top level node includes its module name always

- The rest of nodes include their module names only if that is different from their parent's
- Identities include the module prefix

GET a list of resources:

Request:

While retrieving a list of resources, URI should have Accept header as

Accept:application/vnd.yang.collection+xml

or

Accept:application/vnd.yang.collection+json

Response:

The responses are wrapped in resource type "collection {}"

PUT/POST/PATCH

Request:

- No prefix or module name is required if there is no name collision
- Identities must include the module name
- Other node values do not require prefix or module name

4.2 Authgroup

The following APIs are used to create or retrieve the authentication groups for which the username and password that will be used to access the devices are stored.

GET
https://host:8888/api/running/devices/authgroups - to get available authgroups.

PATCH
https://host:8888/api/running/devices/authgroups/ - to incrementally add the authgroup.

4.2.1 Payload

```
<authgroups>
<group>
<name>default_device</name>
<umap>
<local-user>admin</local-user><!-- - -- - Username used for NCS API
authentication-- - -- - >
<remote-name>admin</remote-name><!-- - -- - Username for Device
authentication-- - -- - >
<remote-password>admin</remote-password><!-- - -- - Password on Device
for user account supplied-- - -- - >
```

<remote-secondary-password>admin</remote-secondary-password><!-- - -- -
used for enable password-- - -- - >
</umap>
</group>
</authgroups>

4.3 Device

The following APIs are used to do CRUD operation of the device.

GET
https://host:8888/api/running/devices/device/{device-name} - to get a device

PUT
https://host:8888/api/running/devices/device/{device-name} - to add device to VTS or replace a device

PATCH
https://host:8888/api/running/devices/device/ - to incrementally add the device (Existing device will not be modified)

4.3.1 Payload to add N9K device

```
<device>
  <name>n9k1</name>
  <address>1.1.1.1</address>
  <authgroup>N9k-authgroup1</authgroup>
  <device-type>
   <cli>
    <ned-id>cisco-nx</ned-id>
    <protocol>ssh</protocol>
   </cli>
  </device-type>
  <ned-settings>
<cisco-nx-connection>
        <method>nxapi</method>
</cisco-nx-connection>
  </ned-settings>
  <state>
    <admin-state>unlocked</admin-state>
    <admin-state-description>changed to xyz</admin-state-description>
  </state>
   <device-info>
      <device-use>LEAF</device-use>
      <bgp-peering>
          <bgp-asn>100</bgp-asn>
```

```
                <loopback-if-num>1</loopback-if-num>
                <loopback-if-ip>10.10.10.0/24</loopback-if-ip>
            </bgp-peering>
        </device-info>
    </device>
```

4.4 Inventory

The following APIs are used to upload / retrieve the inventory of the topology.
GET
https://host:8888/api/running/cisco-vts/devices to get list of device and the connected servers information on each device

PUT
https://host:8888/api/running/cisco-vts/devices/device/{device-name} - to add deviceport-server info to NCS or replace a deviceport-server info for this device

PATCH
https://host:8888/api/running/cisco-vts/devices - to incrementally add the deviceport-server info (Existing device will not be modified)

4.4.1 Payload Example

```
<devices
  xmlns="http://cisco.com/ns/yang/vts"
  xmlns:y="http://tail-f.com/ns/rest"
  xmlns:vts="http://cisco.com/ns/yang/vts">
  <device>
    <name>tb3-tor1</name>
    <ports>
      <port>
        <name>Ethernet1/1</name>
        <connection-type
          xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">vts-ids:server
        </connection-type>
        <servers>
          <server>
            <name>controller</name>
            <type
              xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">
              vts-ids:virtual-server
            </type>
            <interface-name>eth3</interface-name>
            <ip>172.20.100.30</ip>
            <connid>e60792bc-35a3-4785-89ac-55865a0fd237</connid>
          </server>
```

```
        </servers>
      </port>
      <port>
        <name>Ethernet1/2</name>
        <connection-type
          xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">vts-ids:server
        </connection-type>
        <servers>
          <server>
            <name>compute1</name>
            <type
              xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">
              vts-ids:virtual-server
            </type>
            <interface-name>eth3</interface-name>
            <ip>172.20.100.31</ip>
            <connid>9d45feea-770e-4d16-80f7-174e41947636</connid>
          </server>
        </servers>
      </port>
    </ports>
  </device>
</devices>
```

4.4.2 Payload Example with FEX

```
<devices
  xmlns="http://cisco.com/ns/yang/vts"
  xmlns:y="http://tail-f.com/ns/rest"
  xmlns:vts="http://cisco.com/ns/yang/vts">
  <device>
    <name>tb3-tor1</name>
     <ports>
       <port>
        <name>port-channel25</name>
        <connection-type
          xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">vts-ids:fabric
        </connection-type>
        <fexs>
          <fex>
            <fex-id>120</fex-id>
            <ports>
              <port>
                <name>Ethernet120/1/1</name>
                <connection-type
                  xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">
                  vts-ids:server
                </connection-type>
```

```
                <connid>0b4451cf-152e-490c-82f9-e39ffadf29c4</connid>
                <servers>
                  <server>
                    <name>1.0.1.3</name>
                    <type
                      xmlns:vts-ids="http://cisco.com/ns/yang/vts/identities">
                    vts-ids:virtual-server
                    </type>
                    <interface-name>Ethernet1/7</interface-name>
                    <ip>1.0.0.10</ip>
                    <connid>77e76cbe-13e0-4a30-b921-7acfb6d67ac5</connid>
                  </server>
                </servers>
              </port>
            </ports>
          </fex>
        </fexs>
      </port>
    </ports>
  </device>
</devices>
```

Configuring Tenant Information

https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<name>

Tenants	Container for set of VTS Tenants
Tenant	List of VTS Tenants
name	Tenant Name

Configuring Topology Information

https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<id>

Topologies	Container for set of Topologies belonging to a tenant
Topology	List of Topologies
id	Topology Name

4.5 Device-Groups

A device-group can be used to group a set of L2/L3 VTEP devices such that these share a common vlan pool. The VTEPs can be physical or virtual end points.

Once added to the device-group, the device specific vlan pools are no longer used.

Operations: CREATE/UPDATE/GET/DELETE

Create (PUT)

Create a device-group in VTS
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/<device-group-name>

Parameter	Value	Description
Name	String	Name of the device group
Type	String	Type of device group. Default "device-group-type"
vmmid	String	VMM ID
devices	-	Container for list of devices
device/name	String	Name of device in VTS inventory
device/type	String	Type of device – physical or virtual – cisco-vts-identities:p-vtep/ cisco-vts-identities:v-vtep

Sample Payload

https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/tb12-tor-group

```
"device-group": [
    {
      "name": "tb12-tor-group"
      "type": "device-group-type",
      "vmmid": "6BA0C760-E840-40A4-A949-B06DD18E0998",
      "devices": {
        "device": [
          {
            "name": "tb12-leaf1",
                "type": " cisco-vts-identities:p-vtep"
          },
          {
            "name": "tb12-leaf2",
                "type": " cisco-vts-identities:p-vtep"
          }
        ]
      }
    }
  ]
```

Update (PATCH)

Create a device-group in VTS
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group

```
"device-group": [
    {
      "name": "tb12-tor-group"
      "type": "device-group-type",
      "vmmid": "6BA0C760-E840-40A4-A949-B06DD18E0998",
      "devices": {
        "device": [
          {
            "name": "tb12-leaf1",
           "type": " cisco-vts-identities:p-vtep"
          },
          {
            "name": "tb12-leaf2",
            "type": " cisco-vts-identities:p-vtep"
          },
          {
            “name”:”tb12-virtual-leaf1”,
            “type”:” cisco-vts-identities:v-vtep”
          }
        ]
      }
    }
  ]
```

GET
Show device-group details
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/<device-group-name>

LIST (GET)
Show list of all device-groups in the system
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/

DELETE
Delete a device group.
https://172.20.100.126:8888/api/running/cisco-vts/device-groups/device-group/<device-group-name>

4.6 Network

Shows/Lists information for creates, updates and deletes of Network

Create (PUT)

Create a network in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/<id>

Payload

Parameter	Value	Description
admin-state-up	true/false	Admin state of the network up or down
id	Uuid	Network ID
name	String	Network name
shared	Boolean	Shared across all tenants
status	cisco-vts-identities:active/ cisco-vts-identities:build/ cisco-vts-identities:down	Network Status
provider-network-type	cisco-vts-identities:gre/ cisco-vts-identities:trunk/ cisco-vts-identities:vlan/ cisco-vts-identities:vxlan	Network Type
provider-physical-network	String	Physical Network Name
router-external	boolean	External access to Network
Provider-segmentation-id	String	Segmentation ID

Sample Payload

```
{
   "network":
      {
         "id": "14528457-1761-4378-bdaf-1e4b2ff5e999", --
         "admin-state-up": true, --
         "name": "network-1", --
         "provider-physical-network": "physnet1",
         "provider-segmentation-id": "1111",
         "provider-network-type": "cisco-vts-identities:vxlan",
         "router-external": false, --
         "shared": false, --
         "status": " cisco-vts-identities:active", -- tenant name , topology name
```

```
    }
}
```

GET

Show network details associated to one VTS Network under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/<id>

LIST (GET)

Show list of all networks under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/

DELETE

Delete Network under topology <topology-name> of a tenant <tenant-name> and associated resources.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/networks/network/<id>

4.7 Subnet

Shows/Lists information for creates, updates and deletes of Subnet

Create (PUT)

Create a subnet in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/<id>

Parameter	Value	Description
network-id	Uuid	Network ID
Id	Uuid	Subnet ID
Name	String	Subnet name
Shared	Boolean	Shared across all tenants
enable-dhcp	Boolean	True if DHCP is enabled
cidr	String	The CIDR
gateway-ip	String	The gateway IP address
ip-version	String	IP version 4 or 6

Sample Payload:

```
{
 "subnet":
  {
```

```
    "id": "98944325-eb56-56f1-9468-095028d8d36c",
    "network-id": "489ed271-9427-4fa0-85ca-1ca7187d29d1",
    "name": "subnet1",
    "enable-dhcp": true,
    "cidr": "2.2.2.0/24",
    "gateway-ip": "2.2.2.1",
    "ip-version": "4",
    "shared": false
      }
}
```


GET

Show subnet details associated to one VTS Subnet under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/<id>

LIST (GET)

Show list of all subnets under topology <topology-name> of a tenant <tenant-name>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/

DELETE

Delete subnet under topology <topology-name> of a tenant <tenant-name> and associated resources.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/subnets/subnet/<id>

4.8 VM Port

Shows/Lists information for creates, updates and deletes of port

Create (PUT)

Create a port in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/<id>

Parameter	Value	Description
network-id	Uuid	Network ID
id	Uuid	port ID
name	String	Port name
status	cisco-vts-identities:active/ cisco-vts-identities:build/ cisco-vts-identities:down	Network Status

admin-state-up	Boolean	Admin state of the network up or down
connid	Uuid	Connection id of the device

Sample Payload:


```
{
 "port":
 {
        "id": "2d6fa173-9293-408e-8292-7cab94cb9335",
        "network-id": "0266b192-09e7-4bca-b876-e6f62c10d82c",
        "name": "Port2",
        "admin-state-up": true,
        "status": " cisco-vts-identities:active",
        "connid": [{"id": "ff94f7a6-f482-4798-kiu0 960a2cbf9756"}]
 }
}
```


GET

Show port details associated to one VTS Port under topology <topology-name> of a tenant <tenant-name>, belonging to network with id <id>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/<id>

LIST (GET)

Show list of all ports under topology <topology-name> of a tenant <tenant-name> and network with id <id>.
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/

DELETE

Delete port under topology <topology-name> of a tenant <tenant-name> and network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/ports/port/<id>

4.9 Router

Shows/Lists information for creates, updates and deletes of Routers

Create (PUT)

Create a router in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/<id>

Parameter	Value	Description

network-id	uuid	Network ID
id	uuid	Router ID
name	String	Router name
router-gateway	String	Router gateway ip address

Sample Payload:

```
{
  "router": [
    {
      "id": "d8c58e0e-50ce-4ef8-923e-5d053ef38888",
      "status": " cisco-vts-identities:active",
      "name": "router3",
      "router-gateway": "10.1.1.5"
    }
  ]
}
```

GET

Show router details associated to one VTS router under topology <topology-name> of a tenant <tenant-name> belonging to a network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/<id>

LIST (GET)

Show list of all routers under topology <topology-name> of a tenant <tenant-name> belonging to a network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/

DELETE

Delete router under topology <topology-name> of a tenant <tenant-name> belonging to a network with id <id>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/routers/router/<id>

4.10 Router Interfaces

Shows/Lists information for creates, updates and deletes of Interface

Create (PUT)

Create an interface in VTS
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/<id>

Parameter	Value	Description
Id	uuid	Interface ID
subnet-id	uuid	Subnet ID
router-id	uuid	Router ID
ip-address	String	IP address of the interface
port-create	String	Port creation operation, default - none

Sample Payload:

```
{
  "interface": [
    {
      "subnet-id": "3faf211d-1fca-4aff-9c1c-086cda772bae",
      "router-id": "3faf211d-1fca-4aff-9c1c-086cda777777",
      "ip-address": "11.11.11.11",
      "port-create": "vts:none"
    }
  ]
}
```

GET

Show interface details associated to one VTS Interface under topology <topology-name> of a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/<id>

LIST (GET)

Show list of all interfaces under topology <topology-name> of a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/

DELETE

Delete interface under topology <topology-name> of a tenant <tenant-name>
https://<vtc-ip>:8888/api/running/cisco-vts/tenants/tenant/<tenant-name>/topologies/topology/<topology-name>/interfaces/interface/<id>

4.11 Infrastructure Policy

The following APIs are used to do CRUD operations on the infrastructure policy.

4.11.1 Administrative Domains

The following APIs are used to perform CRUD operations on administrative domains.

4.11.1.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/admin-domain/{name}

4.11.1.2 Payload fields

Field	Data Type	Description
Name	string	Name of the administrative domain. This field is required.

4.11.1.3 Payload example

```
{
  “name”: “my-admin-domain”
}
```

4.11.2 Layer 2 Gateway Groups

The following APIs are used to perform CRUD operations on layer 2 gateways groups.

4.11.2.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l2-gateway-groups/l2-gateway-group{name2}

4.11.2.2 Payload fields

Field	Data Type	Description
Name	string	Name of the L2 gateway group. This field is required.

4.11.2.3 Payload example

```
{
  “name”: “my-L2”
}
```

4.11.3 Devices in a Layer 2 Gateway Groups

The following APIs are used to perform CRUD operations on devices belonging to a layer 2 gateways group.

4.11.3.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group/{name2}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l2-gateway-groups/l2-gateway-group/{name2}/devices/device/{name3}

4.11.3.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as L2 gateway. This field is required.

4.11.3.3 Payload example

```
{
  “name”: “my-l2-device”
}
```

4.11.4 Policy Parameters for a Layer 2 Gateway Group

The following APIs are used to perform CRUD operations on policy parameters for a layer 2 gateways group.

4.11.4.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group/{name2}/ policy-parameters

4.11.4.2 Payload fields

Field	Data Type / Value	Description
distribution-mode	“decentralized-l2”	Mode of distribution
control-plane-protocol	“bgp-evpn” or “flood-and-learn”	Control plane protocol used to learn end point addresses.
arp-suppression	Flag	Enables BGP EVPN ARP suppression.
packet-replication	“multicast” or “ingress-replication”	Packet replication mechanism. It

		determines how packets sent to multiple recipients are replicated, so that each recipient gets a copy.

4.11.4.3 Payload example

```
{
  “distribution-mode”: “decentralized-l2”,
  “arp-suppression”: null
}
```

4.11.5 Parent of a Layer 2 Gateway Group

The following APIs are used to perform CRUD operations to assign a parent group to a layer 2 gateways group.

4.11.5.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l2-gateway-groups/l2-gateway-group/{name2}

4.11.5.2 Payload fields

Note that the fields below are mutually exclusive. That is, only one of them can be included in a request (or response) to VTS.

Field	Data Type	Description
ad-l3-gw-parent	string	This L2 group's L3 gateway parent in the current administrative domain
l3-gw-parent	string	This L2 group's L3 gateway parent in the infrastructure policy hierarchy
l3-dc-gw-parent	string	This group's L3 datacenter gateway parent in the infrastructure policy hierarchy.

4.11.5.3 Payload example

```
{
  “ad-l3-gw-parent”: “my-l3-group”
}
```

4.11.6 Layer 3 Gateway Groups in an Administrative Domain

The following APIs are used to perform CRUD operations on layer 3 gateways groups in an administrative domain.

4.11.6.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l3-gateway-groups/l3-gateway-group{name2}

4.11.6.2 Payload fields

Field	Data Type	Description
Name	string	Name of the L3 gateway group. This field is required.

4.11.6.3 Payload example

```
{
  “name”: “my-l3-group”
}
```

4.11.7 Devices in a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations on devices belonging to a layer 3 gateways group in an administrative domain.

4.11.7.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group/{name2}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{domain}/l3-gateway-groups/l3-gateway-group/{name2}/devices/device/{name3}

4.11.7.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as L3 gateway in an administrative domain. This field is required.

4.11.7.3 Payload example

```
{
  “name”: “my-l3-device”
```

}

4.11.8 Policy Parameters for a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations on policy parameters for a layer 3 gateways group in an administrative domain.

4.11.8.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group/{name2}/policy-parameters

4.11.8.2 Payload fields

Field	Data Type	Description
distribution-mode	"cisco-vts-identities: decentralized-l3" or "cisco-vts-identities: centralized-l3"	Mode of distribution
control-plane-protocol	"cisco-vts-identities: bgp-evpn" or "cisco-vts-identities: flood-and-learn"	Control plane protocol used to learn end point addresses.
arp-suppression	Flag	Enables BGP EVPN ARP suppression.
packet-replication	"cisco-vts-identities: multicast" or "cisco-vts-identities: ingress-replication"	Packet replication mechanism. It determines how packets sent to multiple recipients are replicated, so that each recipient gets a copy.

4.11.8.3 Payload example

```
{
  "packet-replication": "cisco-vts-identities:multicast"
}
```

4.11.9 Parent of a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations to assign a parent group to a layer 3 gateways group in an administrative domain.

4.11.9.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/admin-domains/administrative-domain/{name}/l3-gateway-groups/l3-gateway-group/{name2}

4.11.9.2 Payload fields

Note that the fields below are mutually exclusive. That is, only one of them can be included in a request (or response) to VTS.

Field	Data Type	Description
l3-dc-gw-parent	string	This L3 group's L3 datacenter gateway parent in the infrastructure policy hierarchy.
l3-dci-parent	string	This L3 group's datacenter gateway interconnect parent

4.11.9.3 Payload example

```
{
  “l3-dc-gw-parent”: “my-dc-group”
}
```

4.11.10 Layer 3 Gateway Groups

The following APIs are used to perform CRUD operations on layer 3 gateways groups.

4.11.10.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group{name}

4.11.10.2 Payload fields

Field	Data Type	Description
Name	string	Name of the L3 gateway group. This field is required.

4.11.10.3 Payload example

```
{
  “name”: “my-other-l3-group”
}
```

4.11.11 Devices in a Layer 3 Gateway Group

The following APIs are used to perform CRUD operations on devices belonging to a layer 3 gateways group.

4.11.11.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}/devices/device/{name2}

4.11.11.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as L3 gateway in an administrative domain. This field is required.

4.11.11.3 Payload example

```
{
  “name”: “my-other-l3-device”
}
```

4.11.12 Policy Parameters for a Layer 3 Gateway Group

The following APIs are used to perform CRUD operations on policy parameters for a layer 3 gateways group.

4.11.12.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}/policy-parameters

4.11.12.2 Payload fields

Field	Data Type	Description
distribution-mode	“decentralized-l3” or “centralized-l3”	Mode of distribution
control-plane-protocol	“cisco-vts-identities: bgp-evpn” or “cisco-vts-identities: flood-and-learn”	Control plane protocol used to learn end point addresses.
arp-suppression	Flag	Enables BGP EVPN ARP suppression.
packet-replication	“cisco-vts-identities: multicast” or “cisco-vts-identities: ingress-replication”	Packet replication mechanism. It determines how packets sent to multiple recipients are replicated, so that each recipient gets a copy.

4.11.12.3 Payload example

```
{
  “distribution-mode”: “decentralized-l2”,
  “arp-suppression”: null
}
```

4.11.13 Parent of a Layer 3 Gateway Group in an Administrative Domain

The following APIs are used to perform CRUD operations to assign a parent group to a layer 3 gateways group in an administrative domain.

4.11.13.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name2}

4.11.13.2 Payload fields

Note that the fields below are mutually exclusive. That is, only one of them can be included in a request (or response) to VTS.

Field	Data Type	Description
l3-dc-gw-parent	string	This L3 group's L3 datacenter gateway parent in the infrastructure policy hierarchy.
l3-dci-parent	string	This L3 group's datacenter gateway interconnect parent

4.11.13.3 Payload example

```
{
  “l3-dci-parent”: “my-dci-group”
}
```

4.11.14 Data Center Gateway Groups

The following APIs are used to perform CRUD operations on data center gateway groups.

4.11.14.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group{name}

4.11.14.2 Payload fields

Field	Data Type	Description
Name	string	Name of the data center

		gateway group. This field is required.

4.11.14.3 Payload example

```
{
  “name”: “my-dc-group”
}
```

4.11.15 Devices in a Data Center Gateway Group

The following APIs are used to perform CRUD operations on devices belonging to a data center gateway group.

4.11.15.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group/{name}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group/{name}/devices/device/{name2}

4.11.15.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as data center gateway This field is required.

4.11.15.3 Payload example

```
{
  “name”: “my-dc-device”
}
```

4.11.16 Policy Parameters for a Data Center Gateway Group

The following APIs are used to perform CRUD operations on policy parameters for a data center gateway group.

4.11.16.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dc-gateway-group/{name}/policy-parameters

4.11.16.2 Payload fields

Field	Data Type	Description
distribution-mode	“decentralized-l3” or “centralized-l3”	Mode of distribution

4.11.16.3 Payload example

```
{
  “distribution-mode”: “decentralized-l3”
```

}

4.11.17 Parent of a Data Center Gateway Group

The following APIs are used to perform CRUD operations to assign a parent group to a data center gateway group.

4.11.17.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-gateway-groups/l3-gateway-group/{name}

4.11.17.2 Payload fields

Field	Data Type	Description
l3-dci-parent	string	This data center gateway group's datacenter gateway interconnect parent

4.11.17.3 Payload example

```
{
  “l3-dci-parent”: “my-dci-group”
}
```

4.11.18 Data Center Interconnect Groups

The following APIs are used to perform CRUD operations on data center interconect groups.

4.11.18.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group{name}

4.11.18.2 Payload fields

Field	Data Type	Description
Name	string	Name of the data center gateway group. This field is required.

4.11.18.3 Payload example

```
{
  “name”: “my-dci-group”
}
```

4.11.19 Devices in a Data Center Gateway Group

The following APIs are used to perform CRUD operations on devices belonging to a data center gateway group.

4.11.19.1 URIs

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group/{name}/devices/device/

https://host:8888/api/running/cisco-vts/infrastructure-policy/l3-dc-gateway-groups/l3-dci-group/{name}/devices/device/{name2}

4.11.19.2 Payload fields

Field	Data Type	Description
Name	string	Name of the device acting as data center gateway This field is required.

4.11.19.3 Payload example

```
{
  “name”: “my-dci-device”
}
```

4.12 Route Reflectors

4.12.1 Route Reflector Mode

The following APIs are used to perform CRUD operations on the route reflector mode.

4.12.1.1 URIs

https://host:8888/api/running/cisco-vts/global-settings

4.12.1.2 Payload fields

Field	Data Type	Description
route-reflector-mode	“cisco-vts-identities: global-rr” or “cisco-vts-identities: in-line-rr”	Determines how the route reflector operate

4.12.1.3 Payload example

```
{
  “route-reflector-mode”: “cisco-vts-identities: global-rr”;
}
```

4.12.2 Global Route Reflectors

The following APIs are used to perform CRUD operations on the set of global route reflectors.

4.12.2.1 URIs

https://host:8888/api/running/cisco-vts/global-settings/global-route-reflectors/global-route-reflector

https://host:8888/api/running/cisco-vts/global-settings/global-route-reflectors/global-route-reflector/{name}

4.12.2.2 Payload fields

Field	Data Type	Description
Name	String	Name of the route reflector

4.12.2.3 Payload example

```
{
  “name”: “my-first-route-reflector”
}
```

4.13 Global Settings

The following APIs are used to do CRUD operations on global settings.

4.13.1 Anycast Gateway Address

The following APIs are used to perform CRUD operations on the common MAC address for all L2 Gateway devices on the Infrastructure Policy hierarchy.

4.13.1.1 URIs

https://host:8888/api/running/cisco-vts/global-settings/anycast-gateway/anycast-gw-address

4.13.1.2 Payload fields

Field	Data Type	Description
anycast-gw-address	MAC address e.g.: AA:BB:CC:DD:EE:FF	Common MAC address for all L2 Gateway devices

4.13.1.3 Payload example

```
{
  “anycast-gw-address”: “00:11:22:33:44:55”;
}
```

4.14 Template API

4.14.1 Create Template

POST	/templates	Create a new configuration template in VTS

Sample Payload:

```
{
  "template":[
    {
      "name":"Sample Template",
      "description":"A sample router template",
      "type":"route",
      "created-on":"2016-01-12T09:05:00",
      "modified-on":"2016-01-12T09:30:00",
      "route-template":{
        "rt-seed":"70000",
        "vrf-name":"Sample VRF name",
        "static-routes":{
          "static-route":[
            {
              "destination":"1.1.1.0/24",
              "next-hop":"2.2.2.0/24"
            },
            {
              "destination":"3.3.3.0/24",
              "next-hop":"4.4.4.4/32"
            }
          ]
        },
        "import-route-targets":{
          "import-route-target":[
            {
              "stitching":"",
              "route-target":"100:1000",
              "route-target-type":"route-target-internal"
            },
            {
              "route-target":"192.168.1.2:3001",
              "route-target-type":"route-target-external"
            }
          ]
        },
        "export-route-targets":{
          "export-route-target":[
            {
              "route-target":"172.27.184.56:12345",
```

```
            "route-target-type":"route-target-internal"
          },
          {
            "stitching":"",
            "route-target":"123:60000",
            "route-target-type":"route-target-external"
          }
        ]
      }
    }
  }
]
}
```


Fields:

Field Name	Possible Value	Mandatory
name	A string with a maximum of 128 characters	Y
description	A string with a maximum of 128 characters	N
Type	In this release, there is only one template type: route	Y
created-on	Date-time information in the format CCYY-MM-DDTHH:MM:SS	N (This will be system generated)
modified-on	Date-time information in the format CCYY-MM-DDTHH:MM:SS	N (This will be system generated)
vrf-name	A string with a maximum of 128 characters	N
rt-seed	An integer value with a range of 1-16777215	N
destination	An IPv4 address with prefix length	Y (if a new static-route element is to be added)
next-hop	An IPv4 address (Note: the prefix length information is optional in this case)	Y (if a new static-route element is to be added)
stitching	Can be either present or not present	N
route-target	Can be of one of the following 4 formats: 1. ASN2:NN4 2. ASN4:NN2	Y (if a new import-route-target/export-route-target element is to be added)

	3. IPv4:NN2 where: NN2 and ASN2 has a range of 1-65535 NN4 and ASN4 has a range of 1-4294967295 IPv4 is an IPv4 address in the dotted decimal format	
route-target-type	Can be of one of the following types: route-target-internal, route-target-external, route-target-both	Y (if a new import-route-target/export-route-target element is to be added)

4.14.2 Updating a Template

*Note: In this case, only the fields specified in the JSON body will be updated. All the existing fields will remain the same.

PATCH	/templates/template/<template_name>	Update a template

Sample Payload:


```
{
  "template":[
    {
      "router-template":{
        "rt-seed":"8000",
        "static-routes":{
          "static-route":[
            {
              "destination":"10.10.10.0/24",
              "next-hop":"20.20.20.0/24"
            }
          ]
        },
        "import-route-targets":{
          "import-route-target":[
            {
              "stitching":"",
              "route-target":"8888:99",
              "route-target-type":"route-target-internal"
            }
          ]
```

```
        }
      }
    }
  ]
}
```


4.14.3 Get List of Templates

GET	/templates	Retrieve list of templates

Sample Payload:


```
{
  "cisco-vts-templates:templates":{
    "template":[
      {
        "name":"SR_RT_Template_1"
      },
      {
        "name":"SR_Template_1"
      },
      {
        "name":"Sample Template"
      },
      {
        "name":"Temp_3"
      },
      {
        "name":"Temp_4"
      }
    ]
  }
}
```


4.14.4 Get a template using name

GET	/templates/template/{template_name>}	Gets details about a particular template

Sample Payload:


```
{
  "cisco-vts-templates:template":{
    "name":"TemplateA",
    "description":"TempA",
```

```
    "type":"cisco-vts-templates:route",
    "created-on":"2016-02-19T23:30:12.128+00:00",
    "modified-on":"2016-02-19T23:30:12.129+00:00",
    "route-template":{
      "static-routes":{
        "static-route":[
          {
            "destination":"1.1.1.0/24",
            "next-hop":"2.2.2.0/24"
          }
        ]
      },
      "import-route-targets":{
        "import-route-target":[
          {
            "route-target":"2.2.2.0:24"
          }
        ]
      },
      "rt-seed":12345
    },
}
```


4.14.5 Delete all templates

DELETE	/templates	Deletes all templates. Note: Template can be deleted if its not attached to Tenant or Router.

4.14.6 Delete a template identified by name

DELETE	/templates/template/{template-name}	Deletes a template identified by name. Note: Template can be deleted if its not attached to Tenant or Router.

4.14.7 Get Template attached to Tenant

GET	/cisco-vts/tenants/tenant/{tenant name}/template-maps/template-map	Lists existing tenant assigned to tenant

Sample Payload:


```
{
  "collection":{
    "cisco-vts:template-map":[
```

```
        {
          "template-type":"route",
          "template-name":"TemplateB"
        }
      ]
    }
}
```

4.14.8 Attach Template to Tenant

PUT	/cisco-vts/tenants/tenant/{tenant name}/template-maps/template-map	Attaches a pre-defined template to a tenant

Sample Payload:

```
{
  "template-map":[
    {
      "template-type":"cisco-vts-templates:route",
      "template-name":"template1"
    }
  ]
}
```

4.14.9 Detach Template From Tenant

DELETE	/cisco-vts/tenants/tenant/{tenant name}/template-maps/template-map/route	Detach assigned template from tenant

4.14.10 Get template attached to Router

GET	/cisco-vts/tenants/tenant/{tenant name}/topologies/topology/{topology id}/routers/router/{router id}/template-maps/template-map	Lists existing template assigned to Router

Sample Payload:

```
{
  "template-map":[
    {
      "template-type":"cisco-vts-templates:route",
      "template-name":"template1"
    }
  ]
}
```

4.14.11 Attach Template to Router

PUT	/cisco-vts/tenants/tenant/{tenant name}/topologies/topology/topology id}/routers/router/{router id}/template-maps/template-map/route	Attaches a pre-defined template to a router

Sample Payload:

```
{
  "template-map":[
    {
      "template-type":"cisco-vts-templates:route",
      "template-name":"template1"
    }
  ]
}
```

4.14.12 Detach Template From Router

DELETE	/cisco-vts/tenants/tenant/{tenant name}/topologies/topology/{topology id}/routers/router/{router id}/template-maps/template-map/route	Detach Assigned Template From Router

4.14.13 Get Template Type

GET	/templates/template/{template name}/type	For a given template, list the type

Sample Payload:

```
{
  "cisco-vts-templates:type": "cisco-vts-template:route"
}
```

4.14.14 Get List of Import Route Targets

GET	/templates/template/{template-name}/route-template/import-route-targets/import-route-target	For a given template, get list of import route targets

Sample Payload:

```
{
  "cisco-vts-templates:import-route-target":[
    {
      "route-target":"65001:23",
      "route-target-type":"cisco-vts-templates:route-target-internal"
    }
  ]
}
```

4.14.15 Get List of Export Route Targets

GET	/templates/template/{template-name}/route-template/export-route-targets/export-route-target	For a given template, get list of export route targets

Sample Payload:

```
{
  "cisco-vts-templates:export-route-target":[
    {
      "route-target":"65004:22",
      "route-target-type":" cisco-vts-templates:route-target-internal"
    }
  ]
}
```

4.14.16 Set Import Route Target to Template

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target,route target type}	For a given template, add a import route target

Sample Payload:

```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"route-target-external"
    }
  ]
}
```

4.14.17 Set Export Route Target to Template

PUT	/templates/template/{template-name}/route-template/export-route-targets/export-route-target/{route target,route target type}	For a given template, add an export route target

Sample Payload:

```
{
  "export-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"route-target-external"
    }
  ]
}
```

4.14.18 Set Route Target to Internal Device

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target,route target type}	For a given template, add a route target to a ToR

Sample Payload:

```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"route-target-internal"
    }
  ]
}
```

4.14.19 Set Route Target to External Device

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target,route target type}	For a given template, add a route target to a DCI

Sample Payload:


```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":”route-target-external"
    }
  ]
}
```


4.14.20 Set Route Target to Both Internal and External Device

PUT	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/{route target,route target type}	For a given template, add a route target to both a ToR and DCI

Sample Payload:


```
{
  "import-route-target":[
    {
      "route-target":"65004:23",
      "route-target-type":"route-target-both"
    }
  ]
}
```


4.14.21 Get List of Static Routes

GET	/templates/template/{template-name}/route-template/static-routes	For a given template, get list of export route targets

Sample Payload:


```
{
  "cisco-vts-templates:static-routes":{
    "static-route":[
      {
        "destination":"1.1.1.0/24",
        "next-hop":"2.2.2.0/24"
      },
      {
        "destination":"10.10.10.0/24",
        "next-hop":"20.20.20.0/24"
      }
```

```
    ]
  }
}
```

4.14.22 Set a static route to a template

PATCH	/templates/template/{template-name}/route-template/static-routes/static-route/	For a given template, add a static route

Sample Payload:

```
{
    "static-route":[
      {
        "destination":"8.8.8.8/32",
        "next-hop":"4.4.4.4"
      }
    ]
}
```

4.14.23 Delete a specific static route from Template

DELETE	/templates/template/{template-name}/route-template/static-routes/static-route/”<destination_to_be_deleted>,<next_hop_to_be_deleted>”	Delete a specific static route in an existing template

4.14.24 Delete a specific import route target from template

DELETE	/templates/template/{template-name}/route-template/import-route-targets/import-route-target/"<route_target_to_be_deleted, route-target-type>"	Delete a specific import route target in an existing template

4.14.25 Delete a specific export route target from template

DELETE	/templates/template/{template-name}/route-template/export-route-targets/export-route-target/"<route_target_to_be_deleted,route-target-type>"	Delete a specific export route target in an existing template

4.15 Vcenter VTS Plugin API

Current VTS-VCenter plugin supports network provisioning operations through VCenter GUI extensions.

NOTE: The user has to log into vSphere WebClient and enter NCS credentials on the VTS plugin, before any of the rest calls are made.

Base URI:
https://<vCenterIP>:9443/eventMonitorDaemon/EventMonitorWebService

Headers

1.) Create/POST – Content-type – "application/json"
2.) Delete – Content-type – "text/plain"
3.) Authentication - (Key –Authorization, Value – (vc-username/password))

4.15.1 Resource: Network

Create

POST	/network/		Creates a new network

Sample Payload:


```
{"network": {"name": "TestNetwork1",
 "admin-state-up": "true",
 "provider-physical-network": "vlan",
"provider-network-type": " cisco-vts-identities:vxlan",
 "status": "cisco-vts-openstack-identities:ACTIVE",
"router-external": "false",
```

"tenant-name": "Tenant1",

"dcname": "DatacenterSample"}}

Lists

GET	/network/	Lists all the existing networks

Sample Payload:


```
{
    "network" : [{"admin-state-up":"true",
"name":"net8",
"provider-network-type":"cisco-vts-identities:vxlan",
"provider-physical-network":"vlan",
"router-external":"false",
"status":" cisco-vts-openstack-identities:ACTIVE",
"tenant-id":"tenant1",
"tenant-name":"tenant1"
},
{"admin-state-up":"true",
"name":"net9",
"provider-network-type":" cisco-vts-identities:vxlan",
"provider-physical-network":"vlan",
"router-external":"false",
"status":" cisco-vts-openstack-identities:ACTIVE",
"tenant-id":"tenant1",
"tenant-name":"tenant1",
} ]
}
```


Delete

DELETE	/network/{network_name}?dcname={dcname}	Deletes an existing network given its name and data center name on which it resides.

Show

GET	/network/{network_name}/	Lists details of given network named {network_name}

Sample Payload:

{"network":
{"admin-state-up":"true",
"name":"net8",
"provider-network-type":" cisco-vts-identities:vxlan",
"provider-physical-network":"vlan",
"router-external":"false",
"status":" cisco-vts-openstack-identities:ACTIVE",
"tenant-id":"D32508B0-40FC-27A2-B78F-462A08AB44A3",
"tenant-name":"tenant1",
}}

4.15.2 Resource: Subnet

Create

POST	/subnet/	Creates a new subnet

Sample Payload:

{
"subnet": [{
"cidr": "29.0.0.0/24",
"enable-dhcp": "false",
"gateway-ip": "29.0.0.1",
"ip-version": "4",
"name": "Subnet23",
"network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
"shared": "false",
"tenant-id": "tenant1",
"tenant-name": "tenant1"
}]
}

Lists

GET	/subnet	Lists all the existing subnets

Sample Payload:

{
"subnet": [
{

```
            "cidr": "19.0.0.0/24",
            "enable-dhcp": "false",
            "gateway-ip": "19.0.0.1",
            "id": "58717685-c9d1-4aeb-918e-3e0f37034ab5",
            "ip-version": "4",
            "name": "Subnet13",
            "network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
            "shared": "false",
            "tenant-id": "tenant1",
            "tenant-name": "tenant1"
        },
        {
            "cidr": "29.0.0.0/24",
            "enable-dhcp": "false",
            "gateway-ip": "29.0.0.1",
            "id": "c626fc93-0708-45eb-b578-0edee528736d",
            "ip-version": "4",
            "name": "Subnet23",
            "network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
            "shared": "false",
            "tenant-id": "tenant1",
            "tenant-name": "tenant1"
        }
    ]
}
```


Delete

DELETE	/subnet/{subnet_name}/	Deletes a subnet named {subnet_name}

Show

GET	/subnet/{subnet_name}/	Lists details of given subnet named {subnet_name}

Sample Payload:


```
{
    "subnet": {
        "cidr": "29.0.0.0/24",
        "enable-dhcp": "false",
        "gateway-ip": "29.0.0.1",
        "id": "c626fc93-0708-45eb-b578-0edee528736d",
        "ip-version": "4",
        "name": "Subnet23",
        "network-id": "67f75f3d-5cee-45f3-b327-f32885ef8a60",
```

```
        "shared": "false",
        "tenant-id": "tenant1",
        "tenant-name": "tenant1"
    }
}
```


4.15.3 Resource: Router

Create

POST	/router	Creates a new router

Sample Payload:


```
{ "router" : [{
   "status" : "ACTIVE",
   "name" : "Router100",
   "tenant-id" : "Tenant1",
   "vni_number" : 5170,
   "router-gateway" : null
}]
}
```


Lists

GET	/router	Lists all the existing routers

Sample Payload:


```
{"router": [
      {
         "id": "4b4bb642-3238-4d31-acb0-56e90a19cf01",
         "name": "Router2",
         "status": " cisco-vts-identities:ACTIVE",
         "tenant-id": "tenant1",
         "vni_number": "12007"
      },
      {
         "id": "4d602cb8-4f93-43d1-87e0-fd943f18642e",
         "name": "Router5",
         "status": " cisco-vts-identities:ACTIVE",
         "tenant-id": "tenant1",
         "vni_number": "12002"
      }
   ]
}
```


Delete

DELETE	/router/{router_name}/	Deletes a router named {router_name}

Show

GET	/router/{router_name}/	Lists details of given router named {router_name}

Sample Payload:


```
{
   "router": {
      "id": "4d602cb8-4f93-43d1-87e0-fd943f18642e",
      "name": "Router5",
      "status": " cisco-vts-identities:ACTIVE",
      "tenant-id": "tenant1",
      "vni_number": "12002"
   }
}
```


4.15.4 Resource: Interface

Create

POST	/interface	Attach Interface to the router

Sample Payload:


```
{"interfaces" : [{
   "subnet-id": "aabc0bbf-1c63-46a7-99b6-8c7fa91a9ae6",
   "router-id": "aac12354-3573-4472-9370-5e2726bc27eb",
   "ip-address": "46.0.0.1",
   "port-create": "l3:none"
   }]
}
```


Lists

GET	/interface	Lists all the existing interfaces

Sample Payload:


```
{
    "interfaces": [
   {
   "subnet-id": "114d1323-2458-413f-860d-20bed461fc0b",
```

```
"router-id": "04d014f7-a799-43c6-9443-d075495a2e3c",
"ip-address": "29.0.0.1",
"port-create": "l3:none"
},
{
"subnet-id": "909953fc-9b5c-495d-b428-5464d969101b",
"router-id": "04d014f7-a799-43c6-9443-d075495a2e3c",
"ip-address": "14.0.1.1",
"port-create": "l3:none"
}
]
}
```


Show

GET	/interface/{subnet_name}/	Lists details of given interface with subnet name as {subnet_name }

Sample Payload:

{“interfaces” : [{
```
"subnet-id": "114d1323-2458-413f-860d-20bed461fc0b",
"router-id": "04d014f7-a799-43c6-9443-d075495a2e3c",
"ip-address": "29.0.0.1",
"port-create": "l3:none"
}]
```
}

Delete

DELETE	/interface/{subnet_name}/	Detaches subnet named {subnet_name} from router

External Network Attach

POST	/interface/{router_name}/{external_name}?action=attach	Attaches external network named {external_name} to router {router_name}

External Network Detach

POST	/interface/{router_name}/{external_name}?action=detach	Detaches external network named {external_name} to router {router_name}

5 Debugging and troubleshooting

5.1 VTS

5.1.1 Services

There are 3 VTS specific services on VTC :

1) ncs:. The controller and handles NB API etc.
2) tomcat7: Handles the user interface
3) vtsWebServer: Handles Auto Discovery API and Openstack Plugin installation etc.

These services are managed via the Linux service commands, e.g. “service <service-name> status” to query a service.

5.1.2 Logs

The VTC logs are collected in the /var/log/ncs directory

6 Appendix

This appendix explains the YANG model for the VTS REST API.

6.1 VTS Types

```
module cisco-vts-types {

  namespace "http://cisco.com/ns/yang/vts/types";

  prefix vts-types;
  import tailf-common {prefix tailf;}

  organization "Cisco Systems, Inc.";

  contact
```

```
   "Cisco Systems, Inc.
    Customer Service

    Postal: 170 West Tasman Drive
    San Jose, CA 95134

    Tel: +1 800 533-NETS";

  description
   "This module contains a collection of YANG definitions
    for Cisco's VTS's management. Specifically, it defines types used in
    the context of VTS.

    Copyright (c) 2015-2016 by Cisco Systems, Inc.
    All rights reserved.";

  revision "2015-06-13" {
       description
     "Initial revision.";
  }

  /*
   * Typedefs
   */
  typedef uuid {
    type string {
      length "32|36";
    }
  }
  typedef string128 {
    type string {
      length "0..128" {
        error-message "This string cannot exceed 128 characters.";
      }
    }
  }
  typedef non-empty-string128 {
    type string {
      length "1..128" {
        error-message "This string cannot be empty and cannot exceed 128
characters.";
      }
    }
  }
  typedef string255 {
    type string {
      length "0..255" {
```

```
      error-message "This string cannot exceed 255 characters.";
    }
  }
}
typedef string512 {
  type string {
    length "0..512" {
      error-message "This string cannot exceed 512 characters.";
    }
  }
}
typedef string15 {
  type string {
    length "0..15" {
      error-message "This string cannot exceed 15 characters.";
    }
  }
}
typedef vlan-id {
  type uint32 {
    range "2..4094";
  }
}
typedef subinterface-id {
  type uint32 {
    range "1..511";
  }
}
typedef vni {
  type uint32 {
    range "4096..65535";
  }
}
typedef ip-mask {
  type string {
    pattern '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
    + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
    + '/(([0-9])|([1-2][0-9])|(3[0-2]))';
  }
}
typedef vts-multicast-ip {
   type string {
     pattern '(239\.)' +
           '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){2}' +
           '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
  }
 }
```

```
typedef mac {
  type string {
    pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
  }
}

typedef unknown-mac {
  type string {
    pattern 'unknown-.{32,36}';
  }
}

typedef fex-identifier {
  type uint8 {
    range "100..199";
  }
}

typedef asn2-nn4 {
   type string {
      pattern
         '([1-9][0-9]{0,3}|[1-5][0-9]{1,4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-
2][0-9]|6553[0-5])'
         + ':'
         + '([1-9][0-9]{0,8}|199999999[0-8]|[1-3][0-9]{1,9}|4[01][0-9]{8}|42[0-
8][0-9]{7}|429[0-3][0-9]{6}|4294[0-8][0-9]{5}|42949[0-5][0-9]{4}|429496[0-
6][0-9]{3}|4294967[01][0-9]{2}|42949672[0-8][0-9]|429496729[0-5])';
   }
}

typedef asn4-nn2 {
   type string {
      pattern
         '([1-9][0-9]{0,8}|199999999[0-8]|[1-3][0-9]{1,9}|4[01][0-9]{8}|42[0-8][0-
9]{7}|429[0-3][0-9]{6}|4294[0-8][0-9]{5}|42949[0-5][0-9]{4}|429496[0-6][0-
9]{3}|4294967[01][0-9]{2}|42949672[0-8][0-9]|429496729[0-5])'
         + ':'
         + '([1-9][0-9]{0,3}|[1-5][0-9]{1,4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-
2][0-9]|6553[0-5])';
   }
}

typedef asn2-asn2-nn2 {
   type string {
      pattern
```

```
          '([1-9][0-9]{0,3}|[1-5][0-9]{1,4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-
2][0-9]|6553[0-5])'
          + '\.'
          + '([1-9][0-9]{0,3}|[1-5][0-9]{1,4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-
2][0-9]|6553[0-5])'
          + ':'
          + '([1-9][0-9]{0,3}|[1-5][0-9]{1,4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-
2][0-9]|6553[0-5])';
    }
  }

  typedef ipv4-nn2 {
    type string {
      pattern
        '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
        + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
        + ':'
        + '([1-9][0-9]{0,3}|[1-5][0-9]{1,4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-
2][0-9]|6553[0-5])';
    }
  }

  typedef route-target {
    type union{
      type vts-types:asn2-nn4;
      type vts-types:asn4-nn2;
      type vts-types:ipv4-nn2;
    }
    description
      "Three allowed types for route target
        1. ASN2:NN4
        2. ASN4:NN2
        3. IPV4:NN2";
    tailf:info
      "Three allowed types for route target:
        1. ASN2:NN4
        2. ASN4:NN2
        3. IPV4:NN2";
  }

  typedef vpc-identifier {
    description
      "identifier for virtual port channel number";
    type uint16 {
     range "1..4096";
    }
  }
```

```
  typedef vts-assigned-vpc-identifier {
    description
      "used for ether-channel working of VTS";
    type int16 {
     range "-1000..0";
    }
  }

  typedef vpc-id{
    type union{
      type vts-types:vpc-identifier;
      type vts-types:vts-assigned-vpc-identifier;
    }
  }
}
```

6.2 VTS Identities

```
module cisco-vts-identities {

  namespace "http://cisco.com/ns/yang/vts/identities";

  prefix vts-ids;

  import tailf-common {
    prefix tailf;
  }

  organization "Cisco Systems, Inc.";

  contact
    "Cisco Systems, Inc.
    Customer Service

    Postal: 170 West Tasman Drive
    San Jose, CA 95134

    Tel: +1 800 533-NETS";

  description
    "This module contains a collection of YANG definitions
    for Cisco's VTS's management. Specifically, it defines identities used in
    the context of VTS.

    Copyright (c) 2015 by Cisco Systems, Inc.
```

```
   All rights reserved.";

revision "2015-06-13" {
     description
    "Initial revision.";
}

/*
 * Identities
 */
identity network-type {
  description "Based identity from which network types are " +
              "derived.";
}
identity vlan {
  base "network-type";
  description "Vlan network.";
  tailf:info "Vlan network.";
}
identity trunk {
  base "network-type";
  description "Trunk network.";
  tailf:info "Trunk network.";
}
identity gre {
  base "network-type";
  description "GRE network.";
  tailf:info "GRE network.";
}
identity vxlan {
  base "network-type";
  description "VXLAN network.";
  tailf:info "VXLAN network.";
}

identity entity-status {
  description "Based identity from which entities' status are " +
              "derived.";
}
identity active {
  base "entity-status";
  description "Entity is active.";
  tailf:info "Entity is active.";
}
identity down {
  base "entity-status";
  description "Entity is down.";
```

```
  tailf:info "Entity is down.";
}
identity build {
  base "entity-status";
  description "Entity is being built.";
  tailf:info "Entity is being built.";
}

identity binding-type {
  description "Based identity from which port binding types " +
              "are derived.";
}
identity ovs {
  base "binding-type";
  description "Ovs port binding type.";
  tailf:info "Ovs port binding type.";
}
identity unbound {
  base "binding-type";
  description "No binding provided.";
  tailf:info "No binding provided.";
}
identity vhostuser {
  base "binding-type";
  description "Vhostuser port binding type..";
  tailf:info "Vhostuser port binding type.";
}
// TODO:   Consider semantics of a "failed" type. Mind we have port-status.
identity binding-failed {
  base "binding-type";
  description "Port binding failed.";
  tailf:info "Port binding failed.";
}
identity switch-platform {
  description "Based identity from which switch platform types " +
              "are derived.";
}
identity N3K {
  base "switch-platform";
  description "Cisco N3K.";
  tailf:info "Cisco N3K.";
}
identity N5K {
  base "switch-platform";
  description "Cisco N5K.";
  tailf:info "Cisco N5K.";
}
```

```
identity N6K {
  base "switch-platform";
  description "Cisco N6K.";
  tailf:info "Cisco N6K.";
}
identity N7K {
  base "switch-platform";
  description "Cisco N7K.";
  tailf:info "Cisco N7K.";
}
identity N9K {
  base "switch-platform";
  description "Cisco N9K.";
  tailf:info "Cisco N9K.";
}
identity ASR9K {
  base "switch-platform";
  description "Cisco ASR9K.";
  tailf:info "Cisco ASR9K.";
}
identity NOTAPPLICABLE {
  base "switch-platform";
  description "Platform unknown to VTS.";
  tailf:info "Platform unknown to VTS.";
}

identity switch-role {
  description "Based identity from which switch role types " +
              "are derived.";
}
identity leaf {
  base "switch-role";
  description "The device is acting as a leaf.";
  tailf:info "Leaf.";
}
identity spine {
  base "switch-role";
  description "The device is acting as a spine.";
  tailf:info "Spine.";
}
identity border-leaf {
  base "switch-role";
  description "The device is acting as a border-leaf.";
  tailf:info "Border-eaf.";
}
identity dci {
  base "switch-role";
```

```
  description "The device is acting as a DCI.";
  tailf:info "DCI.";
 }
 identity bgp-rr {
  base "switch-role";
  description "The device is acting as a BGP RR.";
  tailf:info "BGP-RR.";
 }
 identity spine-rr {
  base "switch-role";
  description "The device is acting as a spine and also as a RR";
 }
 identity unmanaged {
  base "switch-role";
  description "The device is not managed by VTS";
 }

 identity network-layer {
  description "Based identity from which network layers are " +
              "derived.";
 }
 identity l2 {
  base "network-layer";
  description "Layer 2.";
  tailf:info "Layer 2.";
 }
 identity l3 {
  base "network-layer";
  description "Layer 3.";
  tailf:info "Layer 3.";
 }

 identity l2-mode {
  description "Based identity from which layer 2 modes are " +
              "derived.";
 }
 identity l2-access {
  base "l2-mode";
  description "Access mode.";
  tailf:info "Access.";
 }
 identity l2-trunk {
  base "l2-mode";
  description "Trunk mode.";
  tailf:info "Trunk.";
 }
```

```
identity server-type {
  description "Based identity from which server types are " +
              "derived.";
}
identity baremetal {
  base "server-type";
  description "Baremetal server.";
  tailf:info "Baremetal.";
}
identity virtual-server {
  base "server-type";
  description "Virtual server.";
  tailf:info "Virtual server.";
}
identity fex {
  base "server-type";
  description "Fabrix extender.";
  tailf:info "Fabric extender.";
}

identity vtf {
  base "server-type";
  description "VTF";
  tailf:info "VTF.";
}

identity peer-type {
  description "Based identity from which connection peer types " +
              "are derived.";
}
identity server {
  base "peer-type";
  description "Server.";
  tailf:info "Server.";
}
identity fabric {
  base "peer-type";
  description "Fabric node.";
  tailf:info "Fabric node.";
}


identity port-channel-type {
  description "Based identity from which port channel types " +
              "are derived.";
}
identity vpc {
```

```
    base "port-channel-type";
    description "Virtual port channel type.";
    tailf:info "Virtual port channel.";
  }
  identity no-port-channel {
    base "port-channel-type";
    description "None.";
    tailf:info "None.";
  }

  identity control-plane-protocol {
    description "Base identity from which control plane protocols " +
                "for learning end points' addresses are derived.";
  }
  identity bgp-evpn {
    base "control-plane-protocol";
    description "BGP-EVPN control plane protocol.";
    tailf:info "BGP-EVPN.";
    reference
      "http://www.cisco.com/c/en/us/products/collateral/routers/asr-9000-
series-aggregation-services-routers/whitepaper_c11-731864.pdf";
  }
  identity flood-and-learn {
    base "control-plane-protocol";
    description "The flood-and-learn control plane for learning " +
                "the addresses of end points.";
    tailf:info "Flood-and-learn.";
  }

  identity packet-replication-mode {
    description "Base identity from which modes of packet " +
                "replication are derived. Packet replication is " +
                "required for packets sent to multiple recipients";
  }
  identity multicast {
    base packet-replication-mode;
    description "Packet replication via multicast.";
    tailf:info "Multicast.";
  }
  identity ingress-replication {
    base packet-replication-mode;
    description "Packet replication by ingress-replication, whereby " +
                "a unicast packet is generated per recipient.";
    tailf:info "Ingress replication.";
  }

  identity route-reflector-mode {
```

```
  // TODO: improve description
  description "Base identity from which modes of route " +
              "reflectors.";
}
identity in-line-rr {
  // TODO: improve description
  base route-reflector-mode;
  description "In-line route reflectors.";
  tailf:info "In-line.";
}
identity global-rr {
  // TODO: improve description
  base route-reflector-mode;
  description "Global route reflectors.";
  tailf:info "Global.";
}

identity l3-distribution-mode {
  tailf:info "Base identity from which l3 distribution types are derived.";
}
identity centralized-l3 {
  base l3-distribution-mode;
  description "Centralized L3.";
  tailf:info "Centralized L3.";
}
identity decentralized-l3 {
  base l3-distribution-mode;
  description "Decentralized L3.";
  tailf:info "Decentralized L3.";
}

identity l2-distribution-mode {
  tailf:info "Base identity from which l2 distribution types are derived.";
}
identity decentralized-l2 {
  base l2-distribution-mode;
  description "Decentralized L2.";
  tailf:info "Decentralized L2.";
}

identity op-type {
  tailf:info "Base identity from which operation types are derived.";
}
identity create {
  base op-type;
  description "Create operation.";
  tailf:info "Create operation.";
```

```
  }
  identity delete {
    base op-type;
    description "Delete operation.";
    tailf:info "Delete operation.";
  }

  identity transaction-type {
    tailf:info "Base identity from which transaction types are derived.";
  }
  identity port {
    base transaction-type;
    description "Port operation.";
    tailf:info "Port operation.";
  }
  identity interface {
    base transaction-type;
    description "Interface operation.";
    tailf:info "Interface operation.";
  }

  identity device-group-type {
    description "Based identity from which the device group types are derived.";
  }
  identity dvs {
    base "device-group-type";
    description "DVS.";
    tailf:info "DVS.";
  }

  identity device-group-device-type {
    description "Based identity from which the device in the device group types
are derived.";
  }
  identity v-vtep {
    base "device-group-device-type";
    description "Virtual device type for VTF.";
    tailf:info "Virtual device type for VTF.";
  }
  identity p-vtep {
    base "device-group-device-type";
    description "Physical device type for TOR.";
    tailf:info "Physical device type for TOR.";
  }
  identity service-policy-mode{
    tailf:info "Base identity from which service policy modes are derived.";
  }
```

```
  identity no-policy-mode{
    base service-policy-mode;
    description "No service policy mode.";
    tailf:info "No service policy mode.";
  }
  identity network-to-epg-mode{
    base service-policy-mode;
    description "Network mapped to Endpoint Group.";
    tailf:info "Network mapped to Endpoint Group.";
  }

  identity setup-state{
    description "Base identity for the states in which a component can be during
setup";
    tailf:info "Base identity for the states in which a component can be during
setup";
  }
  identity setup-state-completed{
    base setup-state;
    description "A state to indicate that component has been setup";
    tailf:info "A state to indicate that component has been setup";
  }
  identity setup-state-initial{
    base setup-state;
    description "A state to indicate that component has not been setup";
    tailf:info "A state to indicate that component has not been setup";
  }
  identity setup-state-skipped{
    base setup-state;
    description "A state to indicate that component setup has been skipped";
    tailf:info "A state to indicate that component setup has been skipped";
  }

  identity vmm-type {
    tailf:info "Base identity for vmm type.";
  }
  identity openstack {
    base "vmm-type";
    tailf:info "An openstack vmm.";
  }
  identity vcenter {
    base "vmm-type";
    tailf:info "A vcenter vmm.";
  }

  identity vmm-version {
    tailf:info "Base identity for vmm version.";
```

```
}
identity openstack-version {
  base "vmm-version";
  tailf:info "Base identity for openstack version.";
}
identity openstack-icehouse {
  base "openstack-version";
  tailf:info "Openstack icehouse version.";
}
identity openstack-juno {
  base "openstack-version";
  tailf:info "Openstack juno version.";
}
identity openstack-kilo {
  base "openstack-version";
  tailf:info "Openstack juno version.";
}
identity openstack-liberty-centos {
  base "openstack-version";
  tailf:info "Openstack Liberty version.";
}
identity vcenter-version {
  base "vmm-version";
  tailf:info "Base identity for vcenter version.";
}
identity vcenter-6 {
  base "vcenter-version";
  tailf:info "Vcenter 6.";
}
identity vcenter-5.5 {
   base "vcenter-version";
   tailf:info "Vcenter 5.5.";
}

identity vmm-registration-state{
  description "Base identity for the vmm registration.";
  tailf:info "Base identity for the vmm registration.";
}
identity vmm-registration-completed{
  base vmm-registration-state;
  description "A state to indicate that vmm registration is complete.";
  tailf:info "A state to indicate that vmm registration is complete.";
}
identity vmm-registration-initial{
  base vmm-registration-state;
  description "A state to indicate that vmm registration has not been started.";
  tailf:info "A state to indicate that vmm registration has not been started.";
```

```
}
identity vmm-registration-failed{
 base vmm-registration-state;
 description "A state to indicate that vmm registration has failed.";
 tailf:info "A state to indicate that vmm registration has failed.";
}
identity vmm-registration-in-progress{
 base vmm-registration-state;
 description "A state to indicate that vmm registration is in progress.";
 tailf:info "A state to indicate that vmm registration is in progress.";
}

identity server-capability {
  description "Base identity for the capability of the server.";
  tailf:info "Base identity for the capability of the server.";
 }
 identity no-virtual-switch {
  base server-capability;
  description "Server has no virtual-switch capability. " +
              "Server can only be connected to physical switches.";
  tailf:info "Server has no virtual-switch capability.";
 }
 identity virtual-switch {
  base server-capability;
  description "Server has virtual-switch capability.";
  tailf:info "Server has virtual-switch capability.";
 }
 identity install-status {
  description "Status of the installation.";
  tailf:info "Status of the installation.";
 }
 identity not-applicable {
  base install-status;
  description "Nothing to be installed.";
  tailf:info "Nothing to be installed.";
 }
 identity vtf-installed {
  base install-status;
  description "VTF installed.";
  tailf:info "VTF installed.";
 }
 identity host-agent-installed {
  base install-status;
  description "Host Agent installed.";
  tailf:info "Host Agent installed.";
 }
 identity vtf-host-agent-installed {
```

```
    base install-status;
    description "VTF and host agent installed.";
    tailf:info "VTF and host agent installed.";
   }
   identity vtf-install-failed {
    base install-status;
    description "VTF installation failed.";
    tailf:info "VTF installation failed.";
   }
   identity host-agent-install-failed {
    base install-status;
    description "Host Agent installation failed.";
    tailf:info "Host Agent installation failed.";
   }
   identity vtf-host-agent-install-failed {
    base install-status;
    description "VTF and host agent installation failed.";
    tailf:info "VTF and host agent installation failed.";
   }
   identity vtf-installation-in-progress {
    base install-status;
    description "VTF installation in progress.";
    tailf:info "VTF installation in progress.";
   }
   identity host-agent-installation-in-progress {
    base install-status;
    description "Host Agent installation in progress.";
    tailf:info "Host Agent installation in progress.";
   }
   identity vtf-host-agent-installation-in-progress {
    base install-status;
    description "VTF and host agent installation in progress.";
    tailf:info "VTF and host agent installation in progress.";
   }
}
```

6.3 VTS

```
module cisco-vts {

 namespace "http://cisco.com/ns/yang/vts";

 prefix vts;


 import tailf-common {
  prefix tailf;
```

```
}

include cisco-vts-common {
  revision-date 2015-02-28;
}
include cisco-vts-system-config {
  revision-date 2015-06-15;
}
import cisco-vts-types {
  prefix vts-types;
}
include cisco-vts-network {
  revision-date 2015-06-12;
}
include cisco-vts-inventory {
  revision-date 2015-06-11;
}
include cisco-vts-custom-template {
  revision-date 2015-06-15;
}
include cisco-vts-infrastructure-policy {
  revision-date 2015-06-21;
}
include cisco-vts-device-operational-data {
  revision-date 2015-06-15;
}

include cisco-vts-uuid-server {
  revision-date 2015-06-23;
}

import tailf-ncs {
  prefix ncs;
}

import ietf-inet-types {
  prefix inet;
}

import ietf-yang-types {
    prefix ietf-yang;
  }

import cisco-vts-identities {
  prefix vts-ids;
}
```

```
import resource-allocator {
  prefix ralloc;
}
import vni-allocator {
  prefix vnialloc;
}

import cisco-vts-templates {
  prefix templates;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2015-06-15" {
  description
    "Enhancements:
     Added cisco-vts container which serves as
     root for all cisco-vts objects.
     Added tenants and toplogies.
     Included cisco-vts-system-config submodule.
     Corrections:
    ";
}

revision "2015-02-28" {
   description
    "Initial revision.";
}

identity tagging-requirement-type {
  description "Based identity from which tagging requirements " +
```

```
            "types are derived.";
}
identity optional {
  base "tagging-requirement-type";
  description "Tagging is optional.";
  tailf:info "Tagging is optional.";
}
identity mandatory {
  base "tagging-requirement-type";
  description "Tagging is mandatory.";
  tailf:info "Tagging is mandatory.";
}

/*
 * Identities
 */
identity network-tag-type {
  description "Based identity from which network tag types are " +
            "derived.";
}
identity vlan-tag {
  base "network-tag-type";
  description "Vlan tagging.";
  tailf:info "Vlan tagging.";
}

identity tag-scope-type {
  description "Based identity from which tag scope types are " +
             "derived.";
}
identity global {
  base "tag-scope-type";
  description "The scope of the tag is global.";
  tailf:info "The scope of the tag is global.";
}
identity port {
  base "tag-scope-type";
  description "The scope of the tag is the port.";
  tailf:info "The scope of the tag is the port.";
}

identity port-creation {
  tailf:info "Base identity from which port creation options are derived.";
}
identity none {
  base "port-creation";
  tailf:info "No port creation.";
```

```
  }
  identity create {
    base "port-creation";
    tailf:info "Create port.";
  }

  identity router-gateway-creation {
    tailf:info "Base identity from which router gateway creation options are
derived.";
  }
  identity unset {
    base "router-gateway-creation";
    tailf:info "No router gateway creation.";
  }
  identity set {
    base "router-gateway-creation";
    tailf:info "Set router gateway.";
  }

  grouping VXLAN-NETWORK-ID {
    choice vxlan-network-identifier {
      case user-assigned {
        leaf vni {
          description "Vni to use for this virtual entity";
          type vts-types:vni;
          must "1 = count(/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range[" +
              "vnialloc:start <= current()/../vni " +
              "and vnialloc:end >= current()/../vni])" {
            error-message "The vni must belong to a vni range.";
            tailf:dependency "/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range";
          }
        }
      } // case user-assigned
      case vts-allocated {
        leaf vts-allocated-vni {
          description "Vni allocated by VTC for this virtual entity. The
                       allocation can be dynamic or statically pre-allocated
                       by the user.";
          type vts-types:vni;
          must "1 = count(/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range[" +
              "vnialloc:start <= current()/../vts-allocated-vni " +
              "and vnialloc:end >= current()/../vts-allocated-vni])" {
            error-message "The vni must belong to a vni range.";
            tailf:dependency "/ralloc:resource-pools/ralloc:vni-pool/vnialloc:range";
          }
        }
      } // case vts_allocated
```

```
  } // choice vxlan_network_identifier
} // grouping VXLAN-NETWORK-ID

grouping MULTICAST-ADDRESS-IDENTIFIER {
 choice multicast-ip-address {
  case user-assigned {
   leaf multicast-address {
    description "Multicast address to use for this virtual entity";
    type vts-types:vts-multicast-ip;
    }
  }// case user-assigned
  case vts-allocated {
   leaf vts-allocated-multicast{
    description "Multicast allocated by VTC for this virtual entity. The
                 allocation can be dynamic or statically pre-allocated
                 by the user.";
    type inet:ipv4-address;
   }
  }//case vts-allocated-multicast
 }//choice multicast-ip-address
}//grouping  MULTICAST-ADDRESS-IDENTIFIER


grouping VTS-PORTS {
 container ports  {
  description "Set of ports.";
  tailf:info "Ports.";
  list port {
   description "List of ports.";
   tailf:info "Port.";
   unique mac-address;
   tailf:secondary-index mac-address {
    tailf:index-leafs "mac-address";
   }

   ncs:servicepoint vts-port-servicepoint;

   key id;
   leaf id {
    description "Port identifier.";
    tailf:info "ID.";
    type vts-types:uuid;
   }
   uses ncs:service-data;

   leaf network-id {
    description "Identifier of the associated network.";
```

```
  tailf:info "ID of the associated network.";
  type leafref {
    path "../../../networks/network/id";
  }
  mandatory "true";
}

uses vts:COMMON-PORT;

leaf binding-vif-type {
  description "Binding VIF type.";
  tailf:info "Binding VIF type.";
  type identityref {
    base "vts-ids:binding-type";
  }
  default vts-ids:ovs;
}
leaf status {
          description "Port status.";
  tailf:info "Status.";
  type identityref {
    base "vts-ids:entity-status";
  }
  mandatory true;
}

choice vlan-network-identifier {
  case user-assigned {
    leaf vlan-id {
      description "Vlan id to use for this port.";
      tailf:info "Vlan for the port.";
      type vts-types:vlan-id;
    }
  } // case user-assigned
  case vts-allocated {
    leaf vts-allocated-vlan-id {
      description "Vlan id allocated by VTC for this port. The allocation can be
                   dynamic or statically pre-allocated by the user.";
      tailf:info "Vlan id allocated for the port.";
      type vts-types:vlan-id;
    }
  } // case vts-allocated
} // choice vlan_network_identifier


//Baremetal-specific fields
leaf type {
```

```
  description "Type of server port.";
  tailf:info "Type of server port.";
  type identityref {
   base "vts-ids:server-type";
  }
  default vts-ids:virtual-server;
 }
 leaf ignore-trunk-vlan {
  description "Flag to ignore L2 config push to Tor ports";
  tailf:info "Flag to ignore L2 config push to Tor ports";
  type boolean;
  default false;
  //when "../type = 'vts-ids:baremetal'";
 }
 leaf tagging {
  description "Indicates whether a baremetal server tags " +
            "the packets it sends.";
  tailf:info "Indicates whether a baremetal server tags " +
           "the packets it sends.";
  type identityref {
   base "tagging-requirement-type";
  }
  //when "../type = 'vts-ids:baremetal'";
 }
 leaf network-tag {
  description "Type of network tag used by a baremetal " +
            "server.";
  tailf:info "Type of network tag used by a baremetal " +
            "server.";
  type identityref {
   base "network-tag-type";
  }
  default vlan-tag;
  //when "../type = 'vts-ids:baremetal'";
 }
 leaf tag-scope {
  description "Scope of the network tag used by a baremetal " +
            "server.";
  tailf:info "Scope of the network tag used by a baremetal " +
           "server.";
  type identityref {
   base "tag-scope-type";
  }
  default global;
  //when "../type = 'vts-ids:baremetal'";
 }
```

```
  container fixed-ips {
    description "Set of fixed IP addresses.";
    tailf:info "IP addresses.";

    list fixed-ip {
      // TODO: this list needs to be revisited
      // 1) is the IP address mandatory?
      // 2) subnet-id comes from OS as a list of subnets
      //    our template concatenates them into the subnet-id leaf.
      //    How is this used? Needs to be revisited
      description "List of fixed IP addresses.";
      tailf:info "Fixed IP address.";
      key ip-address;

      leaf ip-address {
        description "IP address.";
        tailf:info "IP address.";
        type inet:ip-address;
      }
      leaf subnet-id {
        description "Subnetwork identifier.";
        tailf:info "Subnetwork ID.";
        type vts-types:string128;
      }
    } // list fixed-ips
  } // container fixed-ips
  container binding-capabilities {
    description "List of port binding capabilities.";
    tailf:info "Binding capability.";
    leaf port-filter {
      description "Port filter.";
      tailf:info "Port filter.";
      type boolean;
      default "false";
    }
  }
  list connid {
    key id;
    leaf id {
      tailf:info "Unique ID for a ToR, ToR port, server name and server port.";
      type leafref {
        path "/cisco-vts/uuid-servers/uuid-server/connid";
      }
      //type vts-types:uuid;
    }
  } // list connid
} // list port
```

```
  } // container ports
} // grouping VTS_PORTS

grouping VTS-SUBNETWORKS {
  container subnetworks  {
    description "Set of subnetworks.";
    tailf:info "Subnetworks.";

    list subnetwork {
      description "List of subnetworks.";
      tailf:info "Subnetwork.";

      ncs:servicepoint vts-subnet-servicepoint;

      key id;
      unique "name";

      leaf id {
        description "Subnetwork identifier.";
        tailf:info "ID.";
        type vts-types:uuid;
      }
      uses ncs:service-data;

      leaf network-id {
        description "Identifier of the associated network.";
        tailf:info "ID of the associated network.";
        type leafref {
          path "../../../networks/network/id";
        }
        mandatory "true";
      }

      uses COMMON-SUBNETWORK;

      container allocation-pools {
        description "Set of allocation pools of IP addresses.";
        tailf:info "Allocation pools.";
        list allocation-pool {
          description "List of allocation pools of IP addresses.";
          tailf:info "Allocation pool.";
          key start;
          // TODO: add "must" constraint so that pools do not overlap
          leaf start {
            description "Allocation pool's first IP address.";
            tailf:info "Pool's first IP address.";
            type inet:ip-address;
```

```
        }
        leaf end {
          description "Allocation pool's last IP address.";
          tailf:info "Pool's last IP address.";
          type inet:ip-address;
          must "../start <= ../end" {
            error-message "The end of the allocation pool " +
                          "must cannot be less than or " +
                          "equal to the start";
           tailf:dependency "../start";
          }
        }
      }  // allocation-pool
    }  // allocation-pools
    container dns-nameservers {
      description "Set of DNS name servers.";
      tailf:info "DNS name servers.";
      leaf-list dns-nameserver {
        description "List of DNS name servers.";
        tailf:info "DNS name server.";
        type inet:ip-address;
      }  // dns-nameserver
    }  // dns-nameservers
    container host-routes {
      description "Set of host routes.";
      tailf:info "Host routes";
      list host-route {
        description "List of host routes.";
        tailf:info "Host route.";
        key destination;
        leaf destination {
          description "Route's destination.";
          tailf:info "Destination.";
          type tailf:ipv4-address-and-prefix-length;
        }
        leaf nexthop {
          description "Route's nexthop gateway.";
          tailf:info "Nexthop gateway.";
          type inet:ip-address;
        }
      }  // host-route
    }  // host-routes
   } // subnetwork
  } // subnetworks
} // VTS_SUBNETWORKS

grouping VTS-NETWORKS {
```

```
container networks  {
  description "Set of networks.";
  tailf:info "Networks.";

  list network {
    description "List of Networks.";
    tailf:info "Network.";
    ncs:servicepoint vts-network-servicepoint;

    key id;

    leaf id {
      description "Network identifier.";
      tailf:info "ID.";
      type vts-types:uuid;
    }

    uses ncs:service-data;

    leaf bridge-domain-id {
      description "Identifier of the associated network.";
      tailf:info "ID of the associated network.";
      type leafref {
        path "../../../bridge-domains/bridge-domain/id";
      }
      // TODO: Commenting out the mandatory flag for now. This needs
      // to be added back in once the network model is validated
      // and the use case for mandating is confirmed
      // mandatory "true";
    }

    uses COMMON-NETWORK;

    leaf status {
      description "Network status.";
      tailf:info "Status.";
      type identityref {
        base "vts-ids:entity-status";
      }
      mandatory true;
    }

    leaf vni-pool {
      description "Vni pool name.";
      tailf:info "Vni pool name.";
      type vts-types:string128;
      // TODO: consider using a leafref to the pool
```

```
        default "vnipool";
      }
      uses VXLAN-NETWORK-ID {
        refine "vxlan-network-identifier/user-assigned/vni" {
          description "Vni to use for this network.";
          tailf:info "Vni for the network.";
        }
        refine "vxlan-network-identifier/vts-allocated/vts-allocated-vni" {
          description "Vni allocated by VTC for this network. The
                       allocation can be dynamic or statically pre-allocated
                       by the user.";
        }
      }
      uses MULTICAST-ADDRESS-IDENTIFIER {
        refine "multicast-ip-address/user-assigned/multicast-address" {
          description "Multicast ip address to use for this network.";
          tailf:info "Multicast ip address for the network.";
        }
        refine "multicast-ip-address/vts-allocated/vts-allocated-multicast" {
          description "Multicast ip address allocated by VTC for this network. The
                       allocation can be dynamic or statically pre-allocated
                       by the user.";
        }
      }
    } // network
  } // networks
} // grouping VTS-NETWORKS

grouping VTS-BRIDGE-DOMAINS {
  container bridge-domains {
    description "Set of Bridge Domains.";
    tailf:info "Bridge Domains.";

    list bridge-domain {
      description "List of Bridge Domains.";
      tailf:info "Bridge Domain.";

      key id;

      leaf id {
        description "Network identifier.";
        tailf:info "ID.";
        type vts-types:uuid;
      }

    } // bridge-domain
  } // bridge-domains
```

```
} // grouping VTS-BRIDGE-DOMAINS

grouping VTS-ROUTER-INTERFACES {
  container interfaces {
    list interface {
      description "List of (router) interfaces.";
      tailf:info "Router interfaces.";
      ncs:servicepoint vts-router-interface-servicepoint;
      uses ncs:service-data;

      key subnet-id;

      leaf subnet-id {
        description "Subnetwork identifier.";
        tailf:info "Subnetwork ID.";
        type leafref {
          path "../../../subnetworks/subnetwork/id";
        }
        mandatory "true";
      }
      leaf router-id {
        description "Router identifier.";
        tailf:info "Router ID.";
        type leafref {
          path "../../../routers/router/id";
        }
        mandatory "true";
      }
      leaf ip-address {
        description "Router interface IP address.";
        tailf:info "IP address.";
        type inet:ip-address;
      }
      leaf port-create {
        description "Port create operation";
        tailf:info "Port create operation";
      //type string;
        type identityref {
          base "port-creation";
        }
        default "none";
      }
      leaf router-gateway-set {
        description "Router gateway set operation";
        tailf:info "Router gateway set operation";
        type identityref {
          base "router-gateway-creation";
```

```
        }
        default "unset";
       }
       leaf router-gateway-ip-address {
        description "Router gateway interface IP address.";
        tailf:info "IP address.";
        type vts-types:ip-mask;
       }
       leaf routerinterface-creation-timestamp {
        type ietf-yang:date-and-time;
        description "The timestamp when the router interface was triggered";
        tailf:info  "The timestamp when the router interface was triggered";
       }

     } // list interfaces
   } // container interfaces
 } // grouping VTS-ROUTER-INTERFACES

 grouping TEMPLATE-MAPS {
   list template-map {
     description "List of VTS configuration template applied.";
     tailf:info "VTS Configuration Template Applied.";

     key "template-type";

     leaf template-type {
       description "Template type.";
       tailf:info "Template Type.";
       type vts-types:string128 {
         pattern "route";
       }
     }

     leaf template-name {
       description "Template name.";
       tailf:info "Template Name.";
       mandatory "true";
       type leafref  {
         path "/templates:templates/templates:template/templates:name";
       }
     }

     leaf last-changed {
       description "A timestamp indicating when template configuration applied
on device has changed.
                    This is used to trigger applying of configuration to devices";
```

```
      tailf:info "A timestamp indicating when template configuration applied on
device has changed. ";
      type ietf-yang:date-and-time;
     }
   } // template-map
  } // grouping TEMPLATE-MAPS

  grouping VTS-ROUTERS {
   container routers {
    description "Set of routers.";
    tailf:info "Routers.";

    list router {
     description "List of routers.";
     tailf:info "Router.";
     uses ncs:service-data;
     ncs:servicepoint vts-router-servicepoint;

     key id;
     unique "name";
     leaf id {
      description "Router identifier.";
      tailf:info "Router ID.";
      type vts-types:uuid;
     }
     leaf name {
      description "Router name.";
      tailf:info "Name.";
      type vts-types:string15;
     }
     leaf router-gateway {
      description "Router gateway.";
      tailf:info "Router gateway.";
      type vts-types:string128;
     }
     leaf router-gateway-ip-address {
      when "../router-gateway";
      description "Router gateway interface IP address.";
      tailf:info "IP address.";
      type vts-types:ip-mask;
     }
     leaf status {
      description "Router status.";
      tailf:info "Router status.";
      type identityref {
       base "vts-ids:entity-status";
      }
```

```
        mandatory true;
       }
       leaf router-creation-timestamp {
        type ietf-yang:date-and-time;
        description "The timestamp when the router was triggered";
        tailf:info  "The timestamp when the router was triggered";
       }

       uses VXLAN-NETWORK-ID {
        refine "vxlan-network-identifier/user-assigned/vni" {
         description "Vni to use for this router";
         tailf:info "Vni for the network.";
        }
        refine "vxlan-network-identifier/vts-allocated/vts-allocated-vni" {
         description "Vni allocated by VTC for this router The
                  allocation can be dynamic or statically pre-allocated
                  by the user.";
        }
       } // uses VXLAN-NETWORK-ID

       container template-maps {
        description "Container for VTS configuration template to router
mappings.";
        tailf:info "VTS Configuration template-router mappings.";
        uses TEMPLATE-MAPS;
       } // container template-maps
     } // list router
   } // container routers
  } // grouping VTS-ROUTERS

  container cisco-vts {
   description "cisco-vts root container.
            This serves as a root for all cisco-vts objects.";
   tailf:info "cisco-vts root container.";


   uses SYSTEM-CONFIG;
   uses VTS-INVENTORY;
   // TODO: enforce in backend code that a device can be in only one vertical in
the infra-policy
   uses INFRA-POLICY;
   uses UUID-SERVER;
   uses VTS-DEVICE-GROUP;

   container tenants {
    description "Set of VTS tenants.";
    tailf:info "Set of VTS tenants.";
```

```
    list tenant {
      description "List of VTS tenants.";
      tailf:info "Tenants.";

      key name;
      leaf name {
        description "Tenant name.";
        tailf:info "Name.";
        type vts-types:string15;
      }

      container template-maps {
        presence "Need to invoke template map service point";
        description "Container for VTS configuration template to tenant
mappings.";
        tailf:info "VTS Configuration template-tenant mappings.";
        uses ncs:service-data;
        ncs:servicepoint template-tenant-map-servicepoint;

        uses TEMPLATE-MAPS;

      } //template-maps

      uses COMMON-VTS-TENANT-GROUPING;

      container topologies {
        description "Set of topologies belonging to a tenant.";
        //Todo: Add an example for topologies
        tailf:info "Set of topologies belonging to a tenant.";

        list topology {
          description "List of topologies.";
          tailf:info "List of topologies.";

          key id;
          leaf id {
            description "Topology name.";
            tailf:info "Topology name.";
            type vts-types:string128;
          }
          uses vts:VTS-BRIDGE-DOMAINS;
          uses vts:VTS-NETWORKS;
          uses vts:VTS-SUBNETWORKS;
          uses vts:VTS-PORTS;
          uses vts:VTS-ROUTERS;
          uses vts:VTS-ROUTER-INTERFACES;
```

```
    } // list topology
   } // container topologies
  } //list tenants
} // container tenant

container vmms{
  list vmm {
   description "Root node for all data required for vmm registration";
   tailf:info "Root node for all data required for vmm registration";

   key id;
   leaf id {
    description "VMM identifier.";
    tailf:info "ID.";
    type vts-types:uuid;
   }

   leaf type {
    description "Type of vmm.";
    tailf:info "Type of vmm.";
    type identityref {
     base "vts-ids:vmm-type";
    }
    mandatory true;
   }

   leaf version {
    description "vmm version.";
    tailf:info "vmm version.";
    type identityref {
     base "vts-ids:vmm-version";
    }
   }

   leaf status {
    description "Registration status.";
    tailf:info "Registration status.";
    type identityref {
     base "vts-ids:vmm-registration-state";
    }
   }

   leaf message {
    description "Detailed message about current registration state.";
    tailf:info "Detailed message about current registration state.";
    type vts-types:string255;
   }
```

```
        leaf userid {
          description "VMM user id";
          tailf:info "VMM user id";
          type string;
        }

        leaf password {
          description "VMM user password.";
          tailf:info "VMM user password";
          type vts-types:string128;
        }

        leaf-list ip-address {
         description "Controller/Network node ip address";
         tailf:info "Controller/Network node ip address";
         type inet:ip-address;
         must "../type = 'vts-ids:openstack' or (../type = 'vts-ids:vcenter' and
count(../ip-address) = 1)" {
          error-message "Only 1 ip-address can be specified for vcenter";
          tailf:dependency "../type";
         }
        }

        leaf port {
         description "host port.";
         tailf:info "host port";
         type uint32;
         when "../type = 'vts-ids:vcenter'";
        }

      } //vmm
    } //vmms
  } //container cisco-vts
}
```

6.4 VTS Common

```
submodule cisco-vts-common {

  belongs-to cisco-vts {
   prefix vts;
  }

  import ietf-inet-types {
   prefix inet;
  }
```

```
import ietf-yang-types {
  prefix yang;
}

import tailf-common {
  prefix tailf;
}
import tailf-ncs {
  prefix ncs;
}
import cisco-vts-types {
  prefix vts-types;
}
import cisco-vts-identities {
  prefix vts-ids;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2015-02-28" {
   description
    "Initial revision.";
}


/*
 * Groupings
 */
grouping COMMON-NETWORK {
  description "Set of common data nodes for virtual networks.";
  leaf admin-state-up {
```

```
    description "The network administrative state is UP.";
    tailf:info "Administrative state up.";
    type boolean;
    default "true";
   }
   leaf name {
    description "Network name.";
    tailf:info "Name.";
     type vts-types:string128;
   }
   leaf provider-physical-network {
    description "Physical network.";
    tailf:info "Physical network.";
     type vts-types:string128;
   }
   leaf provider-segmentation-id {
    description "Segmentation identifier.";
    tailf:info "Segmentation ID.";
     type vts-types:string128;
   }
   leaf provider-network-type {
    description "Network type.";
    tailf:info "Network type.";
    type identityref {
      base "vts-ids:network-type";
    }
   }
   leaf router-external {
    description "External network.";
    tailf:info "External network.";
    type boolean;
    default "false";
   }
   leaf shared {
    description "Shared network.";
    tailf:info "Shared.";
    type boolean;
    default "false";
   }
  }

  grouping COMMON-SUBNETWORK {
   description "Set of common data nodes for virtual subnetworks.";
   leaf name {
    description "Subnetwork name.";
    tailf:info "Name.";
    type vts-types:string128;
```

```
  }
  leaf cidr {
    description "Subnetwork address.";
    tailf:info "Subnetwork address.";
    type tailf:ipv4-address-and-prefix-length;
    mandatory true;
  }
  leaf enable-dhcp {
    description "Enable DHCP.";
    tailf:info "Enable DHCP.";
    type boolean;
    default "true";
  }
  leaf gateway-ip {
    description "Gateway IP address.";
    tailf:info "Gateway IP address.";
    type inet:ip-address;
    mandatory true;
  }
  leaf ip-version {
    description "IP version.";
    tailf:info "IP version.";
    type enumeration {
      enum 4;
      enum 6;
    }
    default 4;
  }
  leaf shared {
    description "Shared subnetwork.";
    tailf:info "Shared.";
    type boolean;
    default "false";
  }
}

grouping COMMON-PORT {
  description "Set of common data nodes for virtual ports.";
  leaf name {
    description "Port name.";
    tailf:info "Name.";
     type vts-types:string128;
  }
  leaf admin-state-up {
    description "The network administrative state is UP.";
    tailf:info "Administrative state up.";
    type boolean;
```

```
  default "true";
 }
 leaf mac-address {
  description "Port MAC address.";
  tailf:info "MAC address.";
  type union {
   type yang:mac-address;
   type vts-types:unknown-mac;
  }
  mandatory true;
 }
 leaf device-id {
  description "Device Identifier.";
  tailf:info "Device Identifier.";
  type vts-types:string128;
 }
 leaf device-owner {
  description "Device owner.";
  tailf:info "Device owner.";
   type vts-types:string128;
 }
 leaf binding-host-id {
  description "Host identifier";
  tailf:info "Host identifier";
  type vts-types:string128;
 }
 // NOTE: security-groups subtree may move in the future.
 //       It is an "extra" in our driver that is not part of
 //       the ML2 driver API. If ML2 adds official support we
 //       likely move this tree to match.
 // TODO: WHAT TO DO W/ SECURITY GROUPS
 list security-groups {
  description "List of security groups.";
  tailf:info "Security group.";
  key id;
  leaf id {
   description "Security group identifier.";
   tailf:info "Identifier.";
   type vts-types:uuid;
  }
  leaf name {
   description "Security group name.";
   tailf:info "Name.";
   type vts-types:string128;
  }
  leaf tenant-id {
   //TODO: do we need this? the leafref to the network may suffice
```

```
  description "Tenant identifier.";
  tailf:info "Tenant ID.";
  type vts-types:string128;
  // TODO: consider moving to leafref
  //type leafref {
  //  path "/tenant:tenant/tenant:tenant-info/" +
  //        "tenant:vmm-tenant-id";
  //}
}
leaf tenant-name {
  // TODO: do we need to name too? The tenant-id may suffice
  description "Tenant name.";
  tailf:info "Tenant name.";
 type vts-types:string128;
}
leaf description {
  description "Security group description.";
  tailf:info "Description.";
  type vts-types:string128;
}
list security-group-rules {
  description "List of security group rules.";
  tailf:info "Security group rule.";
  key id;
  leaf id {
    description "Security group rules identifier.";
    tailf:info "ID.";
    type vts-types:uuid;
  }
  leaf remote-group-id {
    description "Remote group identifier.";
    tailf:info "Remote group ID.";
    type vts-types:string128;
  }
  leaf direction {
    description "Direction.";
    tailf:info "Direction.";
    type vts-types:string128;
    // TODO: reconsider as an identity
  }
  leaf remote-ip-prefix {
    description "Remote IP address prefix.";
    tailf:info "Remote IP address prefix.";
    type inet:ip-prefix;
  }
  leaf protocol {
    description "Protocol.";
```

```
        tailf:info "Protocol.";
        type vts-types:string128;
        // TODO: reconsider as an identity
       }
       leaf ethertype {
        description "Ethertype.";
        tailf:info "Ethertype.";
        type vts-types:string128;
        // TODO: reconsider as an identity
       }
       leaf port-range-max {
        description "Upper bound of the port range.";
        tailf:info "Port range (maximum).";
        type uint32;
       }
       leaf port-range-min {
        description "Lower bound of the port range.";
        tailf:info "Port range (minimum).";
        type uint32;
        must "../port-range-min <= ../port-range-max" {
         error-message "The minimum in the port range cannot be " +
                       "greater than the maximum";
         tailf:dependency "../port-range-max";
        }
       }
       leaf tenant-id {
        description "Tenant identifier.";
        tailf:info "Tenant ID.";
        type vts-types:string128;
       }
       leaf security-group-id {
        description "Security group indentifier.";
        tailf:info "Security group ID.";
      // TODO: reconsider a maximum length for this leaf.
      type string;
       }
      }
     }
    }

    grouping COMMON-PHYSICAL-PORT {
     description "Set of common data nodes for physical ports.";
     leaf name {
      description "Port name.";
      tailf:info "Name.";
      type vts-types:string128;
     }
```

```
    leaf local-mac {
      description "Mac address of the port.";
      tailf:info "Mac address.";
      type yang:mac-address;
    }
    leaf connection-type {
      description "The type of the connection. That is, to " +
                  "what type of device this port is connected " +
                  "to.";
      tailf:info "Type of device this port is connected to.";
      type identityref {
        base "vts-ids:peer-type";
      }
    }
    leaf mode {
      description "Mode of the port.";
      tailf:info "Port mode.";
      type identityref {
        base "vts-ids:network-layer";
      }
    }
  }

  // Device grouping
  grouping VTS-DEVICE-METADATA {
     leaf group-tag {
       description "Group Tagging of Device";
       tailf:info "Group Tag";
       type vts-types:string128;
     }
  }
  // Server groupings
  grouping COMMON-SERVER {
    description "Set of common data nodes for servers.";
    leaf name {
       description "Name of the server (Hostname typically).";
       tailf:info "Server name.";
       must "not(../../../connection-type = 'vts-ids:fabric') or " +
        " (../../../connection-type = 'vts-ids:fabric' and
/ncs:devices/ncs:device[ncs:name = current()])"{
          error-message "In fabric connection-type, valid device should be present.
          Else connection-type should be server or empty.";
          tailf:dependency "../../../connection-type";
        }
       type vts-types:string128;
    }
    leaf type {
```

```
  description "Type of server.";
  tailf:info "Type of server.";
  type identityref {
   base "vts-ids:server-type";
  }
 }
 leaf mac {
  description "Mac address of the server interface " +
              "connected to this switch port.";
  tailf:info "Mac address of the server interface " +
              "connected to this switch port.";
  type yang:mac-address;
 }
 leaf interface-name {
  description "Name of the server interface connected " +
              "to this switch port.";
  tailf:info "Name of the server interface connected " +
              "to this switch port.";
  type vts-types:string128;
 }
 leaf ip {
  description "IP address of the server interface " +
              "connected to this switch port.";
  tailf:info "IP address of the server interface " +
              "connected to this switch port.";
  type inet:ip-address;
 }
 leaf vmm-id {
  description "VMM which manages the server.";
  tailf:info "VMM which manages the server.";
  type leafref {
   path "/cisco-vts/vmms/vmm/id";
  }
 }
 leaf capability {
  description "Capability of the server.";
  tailf:info "Capability of the server. " +
             "physical or virtual.";
  type identityref {
   base "vts-ids:server-capability";
  }
 }
 leaf username {
  description "Username of server.";
  tailf:info "Username of server.";
  type string;
 }
```

```
  leaf password {
    description "Password of server.";
    tailf:info "Password of server.";
    type string;
  }
  leaf install-status {
    description "Status of installation.";
    tailf:info "Status of installation.";
    type identityref {
      base "vts-ids:install-status";
    }
  }
  leaf install-status-message {
    description "Detailed message describing " +
                "status of installation.";
    tailf:info "Detailed message describing " +
               "status of installation.";
    type vts-types:string255;
  }
}

// TODO: If allocators do not go under cisco-vts, move this out
grouping POOL-ALLOCATIONS {
  description "List of allocations from the pool.";
  list allocation {
    description "List of pool allocations.";
    tailf:info "Pool allocation.";
    tailf:cli-suppress-mode;
    key "id";
    leaf id {
      description "Pool allocation identifier.";
      tailf:info "ID.";
      type uint32;
    }
    list owner {
      description "List of owners of the pool allocation.";
      tailf:info "Owner of the pool allocation.";
      key owner_name;
      leaf owner_name {
        description "Name of the owner of the pool allocation.";
        tailf:info "Name.";
        type vts-types:string128;
      }
    }
  }
}
```

```
  grouping COMMON-VTS-TENANT-GROUPING {
    description "Set of common data nodes for vts tenants.";
    leaf description {
      description "Tenant description.";
      tailf:info "Description.";
      type vts-types:string128;
    }
    leaf vmm-tenant-id {
      description "Tenant identifier.";
      tailf:info "ID.";
      type vts-types:string128;
    }
  }
}
```

6.5 UUID Server

```
submodule cisco-vts-uuid-server {

  belongs-to cisco-vts {
    prefix vts;
  }

  import cisco-vts-types {
    prefix vts-types;
  }
  include cisco-vts-common {
    revision-date 2015-02-28;
  }

  import tailf-common {
    prefix tailf;
  }
  import tailf-ncs {
    prefix ncs;
  }

  include cisco-vts-inventory {
    revision-date 2015-06-11;
  }
  import cisco-vts-identities {
    prefix vts-ids;
  }

  organization "Cisco Systems, Inc.";

  contact
```

```
   "Cisco Systems, Inc.
    Customer Service

    Postal: 170 West Tasman Drive
    San Jose, CA 95134

    Tel: +1 800 533-NETS";

  description
   "This module contains a collection of YANG definitions
    for Cisco's VTS's management. Specifically, for the inventory
    of the physical topology.

    Copyright (c) 2015 by Cisco Systems, Inc.
    All rights reserved.";

  revision "2015-06-23" {
    description
     "Initial revision.";
  }

  grouping UUID-SERVER {
    container uuid-servers {
      description "Set of UUID servers";
      tailf:info "Set of UUID servers";
      list uuid-server {

        uses ncs:service-data;

        tailf:info "UUID to Server mapping";

        key "connid";

        leaf connid {
          tailf:info "Unique ID for a tor name, tor port, server name and server
port";
          type vts-types:uuid;
        }

        leaf vpcid {
          tailf:info "VPC ID for this mapping";
          type vts-types:string128;
          mandatory false;
        }

        leaf torname {
          tailf:info "Tor name";
```

```
        type leafref {
          path "/cisco-vts/devices/device/name";
        }
      }

      leaf portname {
        tailf:info "Port name";
        type vts-types:string512;
        must "/cisco-vts/devices/device/ports/port/name or " +
            "/cisco-vts/devices/device/ports/port/fexs/fex/ports/port/name" {
          error-message "portname should be one of the ports/port/name " +
                  " or ports/port/fexs/fex/ports/port/name";
          tailf:dependency "/cisco-vts/devices/device/ports/port/name";
          tailf:dependency "/cisco-
vts/devices/device/ports/port/fexs/fex/ports/port/name";
        }
      }
      leaf server-id {
        tailf:info "ID of server returned by openstack";
        type vts-types:string512;
        must "/cisco-vts/devices/device/ports/port/servers/server/name or " +
            "/cisco-vts/devices/device/ports/port/fexs/fex/fex-id" {
          error-message "server name should be one of the
ports/port/servers/server " +
                  " or ports/port/fexs/fex/name";
          tailf:dependency "/cisco-
vts/devices/device/ports/port/servers/server/name";
          tailf:dependency "/cisco-vts/devices/device/ports/port/fexs/fex/fex-id";
        }
      }
      leaf interface-name {
        tailf:info "Port on which the server is attached";
        type vts-types:string512;
        must "/cisco-vts/devices/device/ports/port/servers/server/interface-
name or " +
            "/cisco-vts/devices/device/ports/port/fexs/fex/ports/port/name" {
          error-message "server interface should be one of the
ports/port/servers/server " +
                  " or ports/port/fexs/fex/ports/port/name";
          tailf:dependency "/cisco-
vts/devices/device/ports/port/servers/server/interface-name";
          tailf:dependency "/cisco-
vts/devices/device/ports/port/fexs/fex/ports/port/name";
        }
      } // interface-name
      leaf server-type {
        description "Type of server.";
```

```
        tailf:info "Type of server.";
        type identityref {
          base "vts-ids:server-type";
        }
       }
     } // list of uuid-server
    }  // container uuid-servers
  } // grouping UUID-SERVER
} // submodule cisco-vts-uuid-server
```

6.6 Infrastructure Policy

```
submodule cisco-vts-infrastructure-policy {

 belongs-to cisco-vts {
  prefix vts;
 }

 import tailf-common {
  prefix tailf;
 }

 import tailf-ncs {
  prefix ncs;
 }

 include cisco-vts-common {
  revision-date 2015-02-28;
 }
 include cisco-vts-inventory {
  revision-date 2015-06-11;
 }
 import cisco-vts-types {
  prefix vts-types;
 }
 import cisco-vts-identities {
  prefix vts-ids;
 }

 organization "Cisco Systems, Inc.";

 contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134
```

```
   Tel: +1 800 533-NETS";

 description
  "This module contains a collection of YANG definitions
   for Cisco's VTS  management. Specifically, for the
   management of the infrastructure policy.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

 revision "2015-06-21" {
  description
    "Initial revision.";
 }

 grouping GROUP-POLICY-PARAMS {
  leaf control-plane-protocol {
    description
      "Control plane protocol used to learn end point addresses.";
    tailf:info "Control plane protocol.";
    type identityref {
      base "vts-ids:control-plane-protocol";
    }
  }
  leaf arp-suppression {
    description
      "Enables BGP EVPN ARP suppression.";
     tailf:info "ARP suppression.";
    type empty;
  }
  leaf packet-replication {
    description
      "Packet replication mechanism. It determines how" +
      "packets sent to multiple recipients are " +
      "replicated, so that each recipient gets a copy.";
    tailf:info "Packet replication.";
    type identityref {
      base "vts-ids:packet-replication-mode";
    } // type identityref
  }  // leaf packet-replication
 } // grouping GROUP-POLICY-PARAMS

 grouping GROUP-POLICY-PARAMS-L3GW {
  container policy-parameters {
    description "Nodes in this container control the " +
                "behavior of an L3 group of " +
```

```
            "the infrastructure policy.";
      tailf:info "Policy parameters.";
      leaf distribution-mode {
       description
        "Mode of distribution.";
       tailf:info "Distribution mode.";
       type identityref {
        base "vts-ids:l3-distribution-mode";
       }
      }
      uses GROUP-POLICY-PARAMS {
       refine "packet-replication" {
        must "count(../../../l3-gateway-group/policy-parameters/packet-
replication != current()/../packet-replication) = 0" {
         error-message "L3 gateway groups within the same administrative " +
                  "domain must use the same packet replication mechanism.";
         tailf:dependency ".";
        }
       } // refine "packet-replication"
      } // uses GROUP-POLICY-PARAMS
     } // container policy-parameters
    } // grouping GROUP-POLICY-PARAMS-L3GW


    grouping INFRA-POLICY {
     container infra-policy {
      container admin-domains {
       description "Set of administrative domains.";
       tailf:info "Administrative domains.";
       list admin-domain {
        description "List of administrative domains.";
        tailf:info "Administrative domain.";
        key name;
        leaf name {
         description "Name of the administrative domain.";
         tailf:info "Name.";
         type vts-types:string128;
        }
        leaf description {
         description "Description of the administrative domain.";
         tailf:info "Description.";
         type vts-types:string128;
        }
        container l2-gateway-groups {
         description "Set of layer 2 gateway groups.";
         tailf:info "Layer 2 gateway groups.";
         list l2-gateway-group {
```

```
            description "List of layer 2 gateway groups.";
            tailf:info "Layer 2 gateway group.";
            key name;
            leaf name {
              description "Name of the L2 gateway group.";
              tailf:info "Name.";
              type vts-types:string128;
                must "count(../../../../admin-domain/l2-gateway-groups/l2-gateway-
group[name = current()/../name]) =1" {
                    error-message "The name of a l2 gateway group must be unique
system wide.";
                    tailf:dependency ".";
                }
            }
            leaf description {
              description "Description of Layer 2 gateway group.";
              tailf:info "Description.";
              type vts-types:string128;
            }
          container devices {
            description "Set of layer 2 devices, acting as layer 2 gateways.";
          tailf:info "Devices.";
          list device {
            uses ncs:service-data;
            ncs:servicepoint l2-gateway-group-servicepoint;
            key name;
            leaf name {
              description "Name of the device.";
              tailf:info "Name.";
              type leafref {
                path "/cisco-vts/devices/device/name";
              }
              must "count(../../../../../../admin-domain/l2-gateway-groups/l2-
gateway-group/devices/device[name = current()/../name]) = 1" {
                error-message "A device can only belong to one l2 gateway group.";
                tailf:dependency ".";
              }
              must "/vts:cisco-vts/vts:global-settings/vts:anycast-
gateway/vts:anycast-gw-address" {
                error-message "L2 gateway requires an anycast gateway mac
address.";
                tailf:dependency "/vts:cisco-vts/vts:global-settings/vts:anycast-
gateway/vts:anycast-gw-address";
              }
            }
          } // list device
        } // container devices
```

```
        choice parent {
          case ad-l3-gw-parent-case {
            leaf ad-l3-gw-parent {
              description "This group's L3 gateway parent in " +
                          "the current administrative domain " +
                          "in the infrastructure policy hierarchy.";
              tailf:info "L3 gateway in current admin domain.";
              type leafref {
                path "../../../l3-gateway-groups/l3-gateway-group/name";
              }
            }
          } // l3-gw-parent-case
          case l3-gw-parent-case {
            leaf l3-gw-parent {
              description "This group's L3 gateway parent in the infrastructure
policy hierarchy.";
              tailf:info "L3 gateway.";
              type leafref {
                path "../../../../../l3-gateway-groups/l3-gateway-group/name";
              }
            }
          } // l3-gw-parent-case
          case l3-dc-gw-parent-case {
            leaf l3-dc-gw-parent {
              description "This group's L3 datacenter gateway parent in the
infrastructure policy hierarchy.";
              tailf:info "L3 datacenter gateway.";
              type leafref {
                path "../../../../../l3-dc-gateway-groups/l3-dc-gateway-
group/name";
              }
            }
          } // case l3-dc-gw-parent-case
        } // choice parent
        container policy-parameters {
          description "Nodes in this container control the " +
                      "behavior of an L2 group of " +
                      "the infrastructure policy.";
          tailf:info "Policy parameters.";
          leaf distribution-mode {
            description
              "Mode of distribution.";
            tailf:info "Distribution mode.";
            type identityref {
              base "vts-ids:l2-distribution-mode";
            }
```

```
            }
            uses GROUP-POLICY-PARAMS {
              refine "packet-replication" {
                must "count(../../../l2-gateway-group/policy-parameters/packet-
replication != current()/../packet-replication) = 0" {
                  error-message "L2 gateway groups within the same administrative "
+
                              "domain must use the same packet replication
mechanism.";
                  tailf:dependency ".";
                }
              }
            } // uses GROUP-POLICY-PARAMS
          } // container policy-parameters
        } // list l2-gateway-groups
      } // container l2-gateway-groups

      container l3-gateway-groups {
        description "Set of layer 3 gateway groups.";
        tailf:info "Layer 3 gateway groups.";
        list l3-gateway-group {
          description "List of layer 3 gateway groups.";
          tailf:info "Layer 3 gateway group.";
          key name;
          leaf name {
            description "Name of the L3 gateway group.";
            tailf:info "Name.";
            type vts-types:string128;
          }
          leaf current-dc-gw-parent {
            description "Name of the current L3 DC gateway group parent chosen
to achieve load balancing.";
            tailf:info "Name of current L3 DC gateway group parent.";
            type vts-types:string128;
          }
          leaf description {
            description "Description of the layer 3 gateway group.";
            tailf:info "Description.";
            type vts-types:string128;
          }
          choice parent {
            case l3-dc-gw-parent-case {
              container l3-dc-gw-parents {
                description "This groups's set of (L3 datacenter gateway group)
parents in the infrastructure policy hierarchy. " +
                            "When multiple parents are defined, they are used for load
balancing. ";
```

```
            tailf:info "L3 datacenter gateway group parents.";

            list l3-dc-gw-parent {
              description "List of this groups's set of L3 datacenter gateway
parents in the infrastructure policy hierarchy.";
              tailf:info "L3 datacenter gateway group parent.";

              key name;
              leaf name {
                description "A L3 datacenter gateway group acting as parent of
this group in the infrastructure policy hierarchy.";
                tailf:info "Name.";
                type leafref {
                  path "../../../../../../../l3-dc-gateway-groups/l3-dc-gateway-
group/name";
                }
              }
            } // list l3-dc-gw-parent
          } // container l3-dc-gw-parents
        } // case l3-dc-gw-parent-case
      } // choice parent

      container devices {
        description "Set of layer 3 devices, acting as site-wide data center
gateways.";
        tailf:info "Devices.";
          list device {
            uses ncs:service-data;
            ncs:servicepoint l3-gateway-group-servicepoint;
            key name;
            leaf name {
            description "Set of layer 3 devices, acting as layer 3 gateways.";
              tailf:info "Name.";

            type leafref {
               path "/cisco-vts/devices/device/name";
              }
              must "count(../../../../../../admin-domain/l3-gateway-groups/l3-
gateway-group/devices/device[name = current()/../name]) = 1" {
                error-message "A device can only belong to one l3 gateway
group.";
                    tailf:dependency ".";
              }
          } // list device
          }
      } // container devices
      uses GROUP-POLICY-PARAMS-L3GW;
```

```
        } // list l3-gateway-groups
       } // container l3-gateway-groups
      } // list admin-domain
    } // container admin-domains

    container l3-gateway-groups {
      description "Set of layer 3 gateway groups.";
      tailf:info "Layer 3 gateway groups.";
      list l3-gateway-group {
        description "List of layer 3 gateway groups.";
        tailf:info "Layer 3 gateway group.";
        key name;
        leaf name {
          description "Name of the L3 gateway group.";
          tailf:info "Name.";
          type vts-types:string128;
        }
        leaf current-dc-gw-parent {
           description "Name of the current L3 DC gateway group parent chosen to
achieve load balancing.";
          tailf:info "Name of current L3 DC gateway group parent.";
          type vts-types:string128;
        }
        leaf description {
          description "Description of the layer 3 gateway group.";
          tailf:info "Description.";
          type vts-types:string128;
        }
        choice parent {
          case l3-dc-gw-parent-case {
            container l3-dc-gw-parents {
              description "This groups's set of (L3 datacenter gateway group) parents
in the infrastructure policy hierarchy. " +
                          "When multiple parents are defined, they are used for load
balancing. ";
              tailf:info "L3 datacenter gateway group parents.";

              list l3-dc-gw-parent {
                description "List of this groups's set of L3 datacenter gateway parents
in the infrastructure policy hierarchy.";
                tailf:info "L3 datacenter gateway group parent.";

                key name;
                leaf name {
                  description "A L3 datacenter gateway group acting as parent of this
group in the infrastructure policy hierarchy.";
                  tailf:info "Name.";
```

```
                type leafref {
                  path "../../../../../l3-dc-gateway-groups/l3-dc-gateway-
group/name";
                }
               }
              } // list l3-dc-gw-parent
             } // container l3-dc-gw-parents
            } // case l3-dc-gw-parent-case
           } // choice parent

         container devices {
          description "Set of layer 3 devices, acting as layer 3 gateways.";
        tailf:info "Devices.";
        list device {
         uses ncs:service-data;
         ncs:servicepoint l3-gateway-group-servicepoint;
         key name;
         leaf name {
          description "Name of the device.";
          tailf:info "Name.";
          type leafref {
           path "/cisco-vts/devices/device/name";
          }
            must "count(../../../../l3-gateway-group/devices/device[name =
current()/../name]) = 1" {
              error-message "A device can only belong to one l3 gateway group.";
              tailf:dependency ".";
             }
            }
          } // list device
        } // container device
        uses GROUP-POLICY-PARAMS-L3GW;
        } // list l3-gateway-groups
       } // container l3-gateway-groups

       container l3-dc-gateway-groups {
        description "Set of layer 3 gateway groups, acting " +
                  "as data center gateways at the site level.";
        tailf:info "Layer 3 gateway (site DCGW) groups.";
        list l3-dc-gateway-group {
         description "List of layer 3 gateway groups, acting " +
                   "as data center gateways. " +
                   "Devices in a group must be configured to provide redundancy
among themselves";
         tailf:info "Layer 3 gateway (site DCGW) group.";
         key name;
         leaf name {
```

```
          description "Name of the L3 gateway group " +
                  "(data center gateway).";
          tailf:info "Name.";
          type vts-types:string128;
        }
        leaf description {
          description "Description of the layer 3 data center gateway group.";
          tailf:info "Description.";
          type vts-types:string128;
        }
        leaf l3-dci-parent {
          description "This group's datacenter gateway interconnect parent in the
infrastructure policy hierarchy.";
          tailf:info "DCI.";
          type leafref {
            path "../../../l3-dci-groups/l3-dci-group/name";
          }
          mandatory true;
        }

        container devices {
       description  "Set of layer 3 devices, acting as site-wide data center
gateways.";
       tailf:info "Devices.";
       list device {
         uses ncs:service-data;
         ncs:servicepoint dcgw-gateway-group-servicepoint;
         description "List of the devices.";
         tailf:info "device.";

         key name;

         leaf name {
           description "Name of the device.";
           tailf:info "Name.";

           type leafref {
             path "/cisco-vts/devices/device/name";
           }

             must "count(../../../../l3-dc-gateway-group/devices/device[name =
current()/../name]) = 1" {
               error-message "A device can only belong to one l3 site-wide data
center gateway group.";
               tailf:dependency ".";
             }
         }
```

```
    }// list device
   }//devices

   container policy-parameters {
    description "Nodes in this container control the " +
                "behavior of an L2 group of " +
                "the infrastructure policy.";
    tailf:info "Policy parameters.";
    leaf control-plane-protocol {
     description
      "Control plane protocol used to learn end point addresses.";
     tailf:info "Control plane protocol.";
     type identityref {
      base "vts-ids:control-plane-protocol";
     }
    } // leaf control-plane-protocol
   } // container policy-parameters
  } // list l3-gateway-site-dcgw-group
 } // container l3-gateway-site-dcgw-groups
 container l3-dci-groups {
  description "Set of layer 3 gateway groups, acting " +
              "as data center interconnects.";
  tailf:info "Layer 3 gateway (DCI) groups.";
  list l3-dci-group {
   description "List of layer 3 gateway groups, acting " +
               "as data center interconnects.";
   tailf:info "Layer 3 gateway (DCI) group.";
   key name;
   leaf name {
    description "Name of the L3 gateway group " +
                "(data center gateway).";
    tailf:info "Name.";
    type vts-types:string128;
   }
   leaf description {
    description "Description of the layer 3 data center interconnect group.";
    tailf:info "Description.";
    type vts-types:string128;
   }

   container devices {
   description "Set of layer 3 devices, acting as data center interconnects.";
   tailf:info "Devices.";
   list device {
    uses ncs:service-data;
    ncs:servicepoint dci-gateway-group-servicepoint;
    key name;
```

```
        leaf name {
            description "Name of the device.";
            tailf:info "Name.";

            type leafref {
           path "/cisco-vts/devices/device/name";
          }

            must "count(../../../../l3-dci-group/devices/device[name =
current()/../name]) = 1" {
           error-message "A device can only belong to one data center
interconnect group.";
             tailf:dependency ".";
            }
          }
         } // list device
        } // container devices
       } // list l3-gateway-dci-group
     } // container l3-gateway-dci-groups
   } // container infra-policy
  }  // grouping INFRA-POLICY
}
```

6.7 Device Extension

6.7.1 N9K

```
//module cisco-vts-n9k-extension {
module n9k-extension {
 namespace "http://cisco.com/ns/yang/services/n9k-extension";
 prefix n9k-extension;

 import ietf-inet-types {
  prefix inet;
 }
 import ietf-yang-types {
  prefix yang;
 }
 import tailf-common {
  prefix tailf;
 }
 import tailf-ncs {
  prefix ncs;
 }
 import cisco-vts {
  prefix vts;
 }
```

```
import cisco-vts-types {
  prefix vts-types;
}
import cisco-vts-identities {
  prefix vts-ids;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the management
   of the Cisco N9K devices.

   Copyright (c) 2015-2016 by Cisco Systems, Inc.
   All rights reserved.";

revision "2015-02-28" {
   description
    "Initial revision.";
}

augment "/ncs:devices/ncs:device" {
   when "ncs:device-type/ncs:cli/ncs:ned-id='cisco-nx-id:cisco-nx'";

container device-info {

   presence "Creates device info";

   // TODO: remove. This is a copy of another leaf in the same container w/ the
same name

   uses vts:VTS-DEVICE-METADATA;
   leaf name {
     tailf:info "Device name ( Switch )";
     type leafref  {
       path "/ncs:devices/ncs:device/ncs:name";
     }
```

```
    }

    leaf platform {
     // TODO: reconsider having a leaf saying this is a N9K on a N9k-specific
extension
     //      consider we may different subtypes of N9K platforms
     description "Device platform.";
     tailf:info "Platform.";
      type identityref {
       base "vts-ids:switch-platform";
      }
    }
    leaf OS {
     description "Operating System.";
     tailf:info "Operating System.";
      type vts-types:string128;
    }
    leaf version {
     description "Operating System version.";
     tailf:info "Operating System version.";
     type string;
    }
    leaf device-use  {
     description "Role of the switch. How it is being used.";
     tailf:info "Usage (role) of the switch.";
     type identityref {
       base "vts-ids:switch-role";
      }
    }
    leaf group-id {
       tailf:info "Group id";
       type string;
    }
    leaf peering-mode {
       tailf:info "Mode in which DC-GW operates";
       type enumeration { enum VRF-PEERING; enum INTEGRATED; }
    }
    leaf vrf-capacity {
       tailf:info "Vrf Capacity of the device";
       default 900;
       type uint32;
    }
    container bgp-peering-info {
     description "BGP peering information.";
     tailf:info "BGP peering information.";
     leaf bgp-asn {
       description "BGP ASN number.";
```

```
    tailf:info "BGP ASN number.";
    type uint32;
   }
   leaf loopback-if-num {
    description "BGP loopback interface number.";
    tailf:info "BGP loopback interface number.";
    type uint16;
   }
   leaf loopback-if-ip {
    description "BGP loopback interface IP address.";
    tailf:info "A.B.C.D/LEN;;IP prefix and network mask " +
               "length in format x.x.x.x/m";
    type inet:ip-prefix;
   }
  }
  list port {
   description "List of switch ports.";
   tailf:info "Switch port.";
   key name;

   uses vts:COMMON-PHYSICAL-PORT;

    leaf port-channel-membership {
     // TODO: remove
     description "Port channel membership.";
     tailf:info "Port channel membership.";
     type vts-types:string128;
      // TODO: consider semantics of this default
      default "none";
    }
    leaf remote-interface-name {
     description "Interface name of the remote node connected " +
                 "to the port.";
     tailf:info "Interface name of the remote node connected " +
                 "to the port.";
     type vts-types:string128;
    }

   leaf remote-mac {
    description "MAC address of the remote node connected " +
                " to the port.";
    tailf:info "Remote MAC address.";
     type yang:mac-address;
   }
    leaf remote-type {
     description "Type of the remote server connected " +
                 "to the port.";
```

```
    tailf:info "Type of the remote server connected " +
               "to the port.";
    type identityref {
     base "vts-ids:server-type";
    }
   }
  list remote-server-id {
   key name;
   leaf name {
    description "Identifier of the remote server connected " +
                "to the port.";
    tailf:info "ID of the remote server connected " +
               "to the port.";
    type vts-types:string128;
   }
  }
 } //port
 container vpc {
  leaf vpc-id {
   tailf:info "<-1000 - 4096>  Domain id.";
   type vts-types:vpc-id;
  }
  container vpc-peer {
   leaf vpc-peer-ip {
    tailf:info "Specify destination ip address of peer switch.";
    type inet:ipv4-address;
   }
   leaf vpc-peer-name {
    tailf:info "Specify peer switch name.";
    type vts-types:string128;
   }
  }
  leaf vpc-peer-link {
   description "Peer link of the virtual port channel.";
   tailf:info "<1-4096>;;Port Channel number.";
   type uint16 {
    range "1..4096";
   }
  }
 } // container vpc
 list port-channel {
  description "List of port channel the switch belongs to.";
  tailf:info "Port channel.";
  key name;
  leaf name {
   description "Port channel name.";
   tailf:info "<1-4096>;;Port Channel number.";
```

```
    // TODO: reconsider this into an integer
    type uint16 {
      range "1..4096";
    }
  }
  leaf vpc-id {
    tailf:info ";;<-1000-4096>  VPC id.";
    type vts-types:vpc-id;

  }
  leaf is-vpc-peer-link {
    description "Interface is a vpc peer link.";
    tailf:info "Interface is a vpc peer link.";
    type boolean;
  }
  leaf type {
    description "Type of port channel.";
    tailf:info "Type.";
    type identityref {
      base "vts-ids:port-channel-type";
    }
  }
  list ports {
    description "Set of ports belonging to the port channel.";
    tailf:info "Port inthe port channel.";
    key name;

    leaf name {
      description "Port name.";
      tailf:info "Port name.";
   type vts-types:string128;
    }
  }
} //port-channel-list

container servers {
  description "Servers connected to the switch.";
  tailf:info "Servers.";

   list server {
    description "List of servers connected to the switch.";
    tailf:info "Server.";
            key server-id;
            leaf server-id {
      description "Server identifier.";
      tailf:info "ID.";
              type vts-types:string128;
```

```
            }
        list nic {
          description "List of network controller interfaces on " +
                      "the server.";
          tailf:info "NIC.";
                key id;
                leaf id {
           description "Identifier of the network controller " +
                       "interface";
           tailf:info "ID.";
                       type vts-types:string128;
                }
                leaf mac {
           // TODO: note how the same data is in multiple places
           description "MAC address of the network controller " +
                       "interface";
           tailf:info "MAC.";
           type leafref {
             path "../../../../port/remote-mac" ;
           }
                }
                leaf ip {
           description "IP address of the network controller " +
                       "interface";
           tailf:info "IP.";
                       type inet:ip-address;
                }
          leaf remote-mac {
                  // TODO: note how the same data is in multiple places
           description "MAC address of the switch port the " +
                       "server NIC is connected to";
           tailf:info "Switch port MAC.";
           type leafref {
             path "../../../../port/local-mac" ;
           }
                }
               } //nic
      } //server
    } //servers
    leaf device-group {
      description "Refers the device-group";
      tailf:info "Refers the device-group";
      type leafref {
        path "/vts:cisco-vts/vts:device-groups/vts:device-group/vts:name";
      }
    } //device-group
  } //device-info
```

```
  } //augment
} //module
```

6.7.2 ASR9K

```
module asr9k-extension {
  namespace "http://cisco.com/ns/yang/services/asr9k-extension";
  prefix asr9k-extension;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import tailf-common {
    prefix tailf;
  }
  import tailf-ncs {
    prefix ncs;
  }
  import cisco-vts {
    prefix vts;
  }
  import cisco-vts-types {
    prefix vts-types;
  }
  import cisco-vts-identities {
    prefix vts-ids;
  }

  organization "Cisco Systems, Inc.";

  contact
    "Cisco Systems, Inc.
     Customer Service

     Postal: 170 West Tasman Drive
     San Jose, CA 95134

     Tel: +1 800 533-NETS";

  description
    "This module contains a collection of YANG definitions
     for Cisco's VTS's management. Specifically, for the management
     of the Cisco ASR9K devices.
```

```
    Copyright (c) 2015-2016 by Cisco Systems, Inc.
    All rights reserved.";

  revision "2015-02-28" {
    description
     "Initial revision.";
  }

  augment "/ncs:devices/ncs:device" {
    when "ncs:device-type/ncs:cli/ncs:ned-id='cisco-ios-xr-id:cisco-ios-xr'";

  container device-info {

    presence "Creates device info";

    // TODO: remove. This is a copy of another leaf in the same container w/ the
same name
    uses vts:VTS-DEVICE-METADATA;
    leaf name {
      tailf:info "Device name ( Switch )";
      type leafref  {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }
    leaf platform {
      // TODO: reconsider having a leaf saying this is a ASR9K on a ASR9k-specific
extension
      //      consider we may different subtypes of ASR9K platforms
      description "Device platform.";
      tailf:info "Platform.";
       type identityref {
        base "vts-ids:switch-platform";
      }
       default vts-ids:ASR9K;
    }
    leaf pending-sync {
      description "Does the device need to be synced?";
      tailf:info "Sync status.";
      type empty;
    }
    leaf OS {
      description "Operating System.";
      tailf:info "Operating System.";
       type vts-types:string128;
    }
    leaf version {
```

```
  description "Operating System version.";
  tailf:info "Operating System version.";
  type string;
 }
 leaf device-use  {
  description "Role of the switch. How it is being used.";
  tailf:info "Usage (role) of the switch.";
  type identityref {
   base "vts-ids:switch-role";
  }
 }
 leaf group-id {
    tailf:info "Group id";
    type string;
 }
 leaf peering-mode {
    tailf:info "Mode in which DC-GW operates";
    type enumeration { enum VRF-PEERING; enum INTEGRATED; }
 }
 leaf vrf-capacity {
    tailf:info "Vrf Capacity of the device";
    default 900;
    type uint32;
 }
 container bgp-peering-info {
  description "BGP peering information.";
  tailf:info "BGP peering information.";
  leaf bgp-asn {
   description "BGP ASN number.";
   tailf:info "BGP ASN number.";
   type uint32;
  }
  leaf loopback-if-num {
   description "BGP loopback interface number.";
   tailf:info "BGP loopback interface number.";
   type uint16;
  }
  leaf loopback-if-ip {
   description "BGP loopback interface IP address.";
   tailf:info "A.B.C.D/LEN;;IP prefix and network mask " +
              "length in format x.x.x.x/m";
   type inet:ip-prefix;
  }
 }
 list port {
  description "List of switch ports.";
  tailf:info "Switch port.";
```

```
  key name;

  uses vts:COMMON-PHYSICAL-PORT;

   leaf port-channel-membership {
    // TODO: remove
    description "Port channel membership.";
    tailf:info "Port channel membership.";
    type vts-types:string128;
     // TODO: consider semantics of this default
     default "none";
  }
  leaf remote-interface-name {
    description "Interface name of the remote node connected " +
                "to the port.";
    tailf:info "Interface name of the remote node connected " +
                "to the port.";
    type vts-types:string128;
   }

  leaf remote-mac {
    description "MAC address of the remote node connected " +
                " to the port.";
    tailf:info "Remote MAC address.";
     type yang:mac-address;
  }
   leaf remote-type {
    description "Type of the remote server connected " +
                "to the port.";
    tailf:info "Type of the remote server connected " +
                "to the port.";
    type identityref {
      base "vts-ids:server-type";
    }
   }
  list remote-server-id {
    key name;
    leaf name {
      description "Identifier of the remote server connected " +
                  "to the port.";
      tailf:info "ID of the remote server connected " +
                  "to the port.";
      type vts-types:string128;
    }
   }
 } //port
 container vpc {
```

```
  leaf vpc-id {
   tailf:info "<-1000 - 4096>  Domain id.";
   type vts-types:vpc-id;
  }
  container vpc-peer {
   leaf vpc-peer-ip {
    tailf:info "Specify destination ip address of peer switch.";
    type inet:ipv4-address;
   }
   leaf vpc-peer-name {
    tailf:info "Specify peer switch name.";
    type vts-types:string128;
   }
  }
  leaf vpc-peer-link {
   description "Peer link of the virtual port channel.";
   tailf:info "<1-4096>;;Port Channel number.";
   type uint16 {
    range "1..4096";
   }
  }
 } // container vpc
 list port-channel {
  description "List of port channel the switch belongs to.";
  tailf:info "Port channel.";
   key name;
  leaf name {
   description "Port channel name.";
   tailf:info "<1-4096>;;Port Channel number.";
   // TODO: reconsider this into an integer
   type uint16 {
    range "1..4096";
   }
  }
   leaf-list ports {
   description "Set of ports belonging to the port channel.";
   tailf:info "Port inthe port channel.";
    type vts-types:string128;
  }
   leaf type {
   description "Type of port channel.";
   tailf:info "Type.";
   type identityref {
    base "vts-ids:port-channel-type";
   }
  }
 } //port-channel-list
```

```
container servers {
  description "Servers connected to the switch.";
  tailf:info "Servers.";

   list server {
    description "List of servers connected to the switch.";
    tailf:info "Server.";
           key server-id;
           leaf server-id {
     description "Server identifier.";
     tailf:info "ID.";
            type vts-types:string128;
           }
    list nic {
     description "List of network controller interfaces on " +
               "the server.";
     tailf:info "NIC.";
            key id;
            leaf id {
      description "Identifier of the network controller " +
                 "interface";
      tailf:info "ID.";
                    type vts-types:string128;
            }
            leaf mac {
      // TODO: note how the same data is in multiple places
      description "MAC address of the network controller " +
                 "interface";
      tailf:info "MAC.";
      type leafref {
       path "../../../../port/remote-mac" ;
      }
            }
            leaf ip {
      description "IP address of the network controller " +
                 "interface";
      tailf:info "IP.";
                    type inet:ip-address;
            }
     leaf remote-mac {
              // TODO: note how the same data is in multiple places
      description "MAC address of the switch port the " +
                 "server NIC is connected to";
      tailf:info "Switch port MAC.";
      type leafref {
       path "../../../../port/local-mac" ;
```

```
            }
                }
              } //nic
     } //server
    } //servers

    leaf device-group {
     description "Refers the device-group";
     tailf:info "Refers the device-group";
     type leafref {
      path "/vts:cisco-vts/vts:device-groups/vts:device-group/vts:name";
     }
    } //device-group
   } //device-info
   } //augment
  } //module
```

6.7.3 Device Extension Infra

```
module device-extension-infra-policy {
 namespace "http://cisco.com/ns/yang/services/device-extension-infra-policy";
 prefix dev-infra-policy;

 import tailf-common {
  prefix tailf;
 }
 import tailf-ncs {
  prefix ncs;
 }

 import cisco-vts {
  prefix vts;
 }

 organization "Cisco Systems, Inc.";

 contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

 description
  "This module contains a collection of YANG definitions
```

```
    for Cisco's VTS's management. Specifically, for the management
    of the NCS devices in conjunction with the infrastructure policy.

    Copyright (c) 2015 by Cisco Systems, Inc.
    All rights reserved.";

  revision "2015-06-21" {
   description
     "Initial revision.";
  }

  augment "/ncs:devices/ncs:device" {
   leaf l2-gateway-group {
     description "Layer 2 gateway group this device belongs to.";
     tailf:info "L2 gateway group.";
     type leafref {
       path "/vts:cisco-vts/vts:infra-policy/vts:admin-domains/vts:admin-
domain/vts:l2-gateway-groups/vts:l2-gateway-group/vts:name";
     }
   } // leaf l2-gateway-group
  } //augment
} //module
```

6.8 Inventory

```
  submodule cisco-vts-inventory {

    belongs-to cisco-vts {
      prefix vts;
    }

    import ietf-inet-types {
      prefix inet;
    }

    import tailf-common {
      prefix tailf;
    }
    import tailf-ncs {
      prefix ncs;
    }
    import cisco-vts-types {
      prefix vts-types;
    }
    import cisco-vts-identities {
      prefix vts-ids;
    }
```

```
include cisco-vts-common {
  revision-date 2015-02-28;
}
include cisco-vts-device-operational-data {
  revision-date 2015-06-15;
}

organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the inventory
   of the physical topology.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2015-06-11" {
 description
    "Initial revision.";
}


grouping VTS-INVENTORY {
  container devices {
    description "Devices that VTS manages as part of the inventory.";
    tailf:info "VTS device inventory.";
    list device {
      description "List of switches with the servers and other " +
                  "devices connected to them.";
      tailf:info "Device (Switch) Port mapping.";
      ncs:servicepoint portserver;
      uses ncs:service-data;
      key name;
      leaf name {
        description "Name of switch that is part of VTS inventory.";
        tailf:info "Device (Switch) name.";
        type leafref  {
```

```
    path "/ncs:devices/ncs:device/ncs:name";
   }
 }
 uses VTS-DEVICE-NETWORK-OPER-DATA;
 container ports {
  description "Device interfaces that VTS manages as part of " +
              "inventory.";
  tailf:info "VTS port inventory.";
  list port {
   description "List of physical portson the switch that VTS " +
               "manages as part of inventory.";
   tailf:info "Device (switch) port.";
   key name;
   uses vts:COMMON-PHYSICAL-PORT;
   choice connnected-entities {
    description "A port/port-channel can have connected to it " +
                "either servers of fabric extenders.";
    case connected-servers {
     description "Servers connected to a port/port-channel.";
     container servers {
      description "Servers that VTS manages as part of inventory.";
      tailf:info "VTS server inventory.";
      list server {
       description "List of servers attached to device interfaces " +
                   "that is managed by VTS.";
       tailf:info "VTS server inventory.";
       key name;
       uses vts:COMMON-SERVER;
       leaf connid {
        tailf:info "Unique ID for port server mapping";
        type vts-types:uuid;
       } // connid

       leaf vtf_ip {
        tailf:info "VTF VM IP address";
        type inet:ip-address;
       }
      } // server
     } // servers
    } // case connected-servers
    case connected-fexs {
     description "Fabrix extenders connected to a port/port-channel.";
     container fexs {
      description "Set of FEXs under a physical port of port channel.";
      tailf:info "Fabric extenders.";
      list fex {
       description "List of FEXs under a physical port or port channel.";
```

```
                tailf:info "Fabric extender.";
                key fex-id;
                leaf fex-id {
                 description "Fex identifier.";
                 tailf:info "Id.";
                 type vts-types:fex-identifier;
                }
                container ports {
                 description "Set of host interfaces on the FEX.";
                 tailf:info "Ports.";
                 list port {
                  description "List of host interfaces on the FEX.";
                  tailf:info "Port.";
                  key name;
                  uses vts:COMMON-PHYSICAL-PORT;
                  leaf connid {
                   description "ID for the port-server mapping.";
                   tailf:info "Unique ID for port server mapping";
                   type vts-types:uuid;
                  } // port connid
                  container servers {
                   description "Set of servers connected to this host interface on
the FEX.";
                   tailf:info "Servers.";
                   list server {
                    description "List of servers connected to this host interface on
the FEX.";
                    tailf:info "Server.";
                    key name;
                    uses vts:COMMON-SERVER;
                    leaf connid {
                     description "ID for the port-server mapping.";
                     tailf:info "Unique ID for port server mapping";
                     type vts-types:uuid;
                    } // connid
                   } // server
                  } // servers
                 } // port
                } // ports
               } // fex
              } // fexs
             } // case connected-fexs
            } // choice connnected-entities
           } // port
          } // ports
         } // device
        } // devices
```

```
container vtfs {
 list vtf {
   description "List of VTF VM ips under this server.
                If this list is non-empty, it indicates that
                VTF VMs have been spawned on this server";
   tailf:info "List of VTF VM ids under this server.";
   ncs:servicepoint vtf-registration-servicepoint;
   uses ncs:service-data;

   key ip;

   leaf ip {
    tailf:info "VTF VM IP address";
    type inet:ip-address;
   }
   leaf local-mac {
    tailf:info "Local MAC address";
    //type yang:mac-address;
type string {
        pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
}

   }
   leaf username {
    tailf:info "Username of VTF instance";
    mandatory true;
    type string;
   }
   leaf password {
    tailf:info "Encrypted password of VTF instance";
    type string;
   }
   leaf gateway-ip {
    description "Gateway/Next Hop IP for the VTF VM";
    tailf:info "Gateway/Next Hop IP for the VTF VM";
    type inet:ip-address;
   }
   leaf binding-host-name {
    description "Compute Host identifier";
    tailf:info "Compute Host identifier";
mandatory true;
type string;
    //type leafref {
    //  path "../../../devices/device/ports/port/servers/server/name";
    //}
   }
```

```
        leaf vpp-client-name {
          description "Optional client name for VPP";
          tailf:info "Optional client name for VPP";
          type vts-types:string128;
        }

        leaf device-group {
          description "Refers the device-group";
          tailf:info "Refers the device-group";
          type leafref {
            path "../../../device-groups/device-group/name";
          }
        } //device-group
        leaf subnet-mask {
          description "Subnet mask.";
          tailf:info "Subnet mask.";
          type inet:ip-address;
        }
        leaf underlay-network {
          description "Name of underlay network portgroup/bridge on " +
                      "binding-host which VTF is attached to.";
          tailf:info  "Name of underlay network portgroup/bridge on " +
                      "binding-host which VTF is attached to.";
          type string;
        }
        leaf tenant-network {
          description "Name of tenant network portgroup/bridge on " +
                      "binding-host which VTF is attached to.";
          tailf:info  "Name of tenant network portgroup/bridge on " +
                      "binding-host which VTF is attached to.";
          type string;
        }
        leaf datastore {
          description "Name of datastore on binding-host.";
          tailf:info "Name of datastore on binding-host.";
          type string;
        }
      } // list vtf
    } // container vtf

    container xrvr-groups {
      list xrvr-group {
        description "List of XRVR device groups, consisting of XRVR's  and the
VTF-VMs attached to them ";
        tailf:info "XRVR to VTF Mapping.";
        ncs:servicepoint xrvr-to-vtf-map-servicepoint;
        uses ncs:service-data;
```

```
      key group-name;
      leaf group-name {
       description "Name of xrvr group that VTF's will belong to. Each VTF
will belong to all XRVR's in the group.";
       tailf:info "XRVR group name.";
       type vts-types:string128;
      }

      container xrvr-devices {
       list xrvr-device {
        description "List of XRVR devices";
        tailf:info "XRVR in Group";
        key name;
        leaf name {
         description "Name of switch that is part of VTS inventory.";
         tailf:info "Device (Switch) name.";
         type leafref  {
          path "/ncs:devices/ncs:device/ncs:name";
         }
        }
       } // list xrvr-device
      } //container xrvr-devices

      container vtfs {
       list vtf {
        description "List of VTF VMs managed by the XRVR group";
        tailf:info "VTF VMs that are managed by the XRVR group";
        key ip;

        leaf ip {
         description "IP address of the VTF instance";
         tailf:info "IP address of the VTF instance";
         type inet:ip-address;
        }
       } // list vtf
      } // container vtfs
     } // list xrvr-group
    } // container xrvr-devices

  } // grouping VTS-INVENTORY

  grouping VTS-DEVICE-GROUP {
   container device-groups {
    description "List of device groups";
    tailf:info "List of device groups";
```

```
    list device-group {
      description "Set of devices";
      tailf:info "Set of devices";
      key name;

      leaf name {
        description "Name of the device-group";
        tailf:info "Name of the device-group";
        type vts-types:string128;
        must "0 = count(/ncs:devices/ncs:device[ncs:name = current()/../name])"
{
          error-message "Device Group name could not be the same as the device
name";
          tailf:dependency "/ncs:devices/ncs:device/ncs:name";
        }
      }

      leaf type {
        description "Type of device group";
        tailf:info "Type of device group";
        type identityref {
          base "vts-ids:device-group-type";
        }
      }

      leaf vmmid {
        description "ID of VMM for device group";
        tailf:info "ID of VMM for device group";
        type vts-types:string128;
      }

      container devices {
        description "List of device in the device group";
        tailf:info "List of device in the device group";

        list device {
          description "device in the device group";
          tailf:info "device in the device group";
          key name;
          ncs:servicepoint vts-device-group-servicepoint;
          uses ncs:service-data;

          leaf name {
            description "Name of the device";
            tailf:info "Name of the device";
            type vts-types:string128;
            must "/ncs:devices/ncs:device[ncs:name = current()] or " +
```

```
            "/vts:cisco-vts/vts:vtfs/vts:vtf[ip = current()]";
          }

          leaf type {
            description "Type of device";
            tailf:info "Type of device";
            type identityref {
              base "vts-ids:device-group-device-type";
            }
          }

        } //list device
      } // container devices
    } // list device-group
  } // container device-groups
 } // grouping VTS-DEVICE-GROUP
}
```

6.9 System Config

```
submodule cisco-vts-system-config {

 belongs-to cisco-vts {
  prefix vts;
 }

 import ietf-inet-types {
  prefix inet;
 }

 import tailf-common {
  prefix tailf;
 }

 import tailf-ncs {
  prefix ncs;
 }

 import cisco-vts-types {
  prefix vts-types;
 }
 import cisco-vts-identities {
  prefix vts-ids;
 }
```

```
organization "Cisco Systems, Inc.";

contact
  "Cisco Systems, Inc.
   Customer Service

   Postal: 170 West Tasman Drive
   San Jose, CA 95134

   Tel: +1 800 533-NETS";

description
  "This module contains a collection of YANG definitions
   for Cisco's VTS's management. Specifically, for the
   management of system level configurations.

   Copyright (c) 2015 by Cisco Systems, Inc.
   All rights reserved.";

revision "2015-06-15" {
   description
    "Initial revision.";
}


grouping SYSTEM-CONFIG {
  container global-settings {
    description "Container for system level configurations";

    leaf route-reflector-mode {
      //TODO: improve this description
      description "Determines how the route reflector operate.";
      tailf:info "Route reflector mode.";
      type identityref {
        base "vts-ids:route-reflector-mode";
      }
    }
    container global-route-reflectors {
      description "Set of global route reflectors.";
      tailf:info "Global route reflectors.";
      when "../route-reflector-mode = 'vts-ids:global-rr'";
      list global-route-reflector {
        uses ncs:service-data;
        ncs:servicepoint global-rr-servicepoint;
        description "List of global route reflectors.";
        tailf:info "Route reflector.";
        key name;
```

```
    leaf name {
      description "Name of the route reflector.";
      tailf:info "Name.";
      type vts-types:string128;
      // TODO: move to leafref to devices when ready
      //type leafref {
      //  path "/cisco-vts/devices/device/name";
      //}
    }
  } // list route-reflector
} // container route-reflectors

container dhcp-servers {
  description "Set of DHCP servers.";
  tailf:info "DHCP servers.";
  list dhcp-server {
    uses ncs:service-data;
    description "List of DHCP servers.";
    tailf:info "DHCP server.";
    key ip-address;

    leaf ip-address {
      description "IP address of the DHCP server";
      tailf:info "IP address.";
      type inet:ipv4-address {
        tailf:info "A.B.C.D;;DHCP Server IP Address";
      }
    }

    leaf vrf-name {
      description "VRF name of the DHCP server";
      tailf:info "VRF name.";
      type vts-types:string128;
    }

  } // list dhcp-server
} // container dhcp-servers

container anycast-gateway {
  description "Common anycast configuration for all gateways.";
  tailf:info "Common anycast gateway.";
  presence "Existence of anycast-gateway configuration";
  uses ncs:service-data;
  ncs:servicepoint vts-anycast-gateway-servicepoint;
  leaf anycast-gw-address {
    description "Common anycast MAC address for all gateways.";
    tailf:info "Common anycast MAC address.";
```

```
        type string {
          pattern '[0-9a-fA-F][02468aceACE](:[0-9a-fA-F]{2}){5}' {
            error-message "The anycast gateway mac address must be a unicast
address.";
          }
        }
      }
    } // container anycast-gateway

    container service-policy-config{
      description "Service Policy Configuration";
      tailf:info "Service Policy Configuration";
      leaf policy-mode {
        description "Specify service policy mode";
        tailf:info "Specify service policy mode";
        default "vts-ids:no-policy-mode";
        type identityref {
          base "vts-ids:service-policy-mode";
        }
      }
    }//container service-policy-config
   } // container system-config
  } // grouping
} // module
```